Sokratis Katsikas
Isaac Agudo (Eds.)

LNCS 8341

Public Key Infrastructures, Services and Applications

10th European Workshop, EuroPKI 2013
Egham, UK, September 2013
Revised Selected Papers

Springer

Lecture Notes in Computer Science 8341

Sokratis Katsikas   Isaac Agudo (Eds.)

Public Key Infrastructures, Services and Applications

10th European Workshop, EuroPKI 2013
Egham, UK, September 12-13, 2013
Revised Selected Papers

Springer

Volume Editors

Sokratis Katsikas
University of Piraeus
Department of Digital Systems
150 Androutsou St.
Piraeus 185 32, Greece
E-mail: ska@unipi.gr

Isaac Agudo
University of Malaga
Department of Computer Science
Campus de Teatinos s/n
29071 Málaga, Spain
E-mail: isaac@lcc.uma.es

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-53996-1 e-ISBN 978-3-642-53997-8
DOI 10.1007/978-3-642-53997-8
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013957062

CR Subject Classification (1998): K.6.5, C.2, E.3, D.4.6, J.1, K.4.4

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at the 10th European Workshop on Public Key Infrastructures, Services and Applications (EuroPKI 2013) held during September 11–12, 2013 in conjunction with ESORICS 2013 in Egham, U.K.

The workshop received 20 submissions. Each submission was subjected to a thorough review by at least three Program Committee members and external reviewers. The papers were evaluated on the basis of their significance, novelty, and technical quality. Reviewing was double-blind meaning that the Program Committee was not able to see the names and affiliations of the authors, and the authors were not told which Committee members reviewed which papers.

These proceedings contain the 11 accepted publications and the presentation paper by the invited speaker Fabio Martinelli.

We wish to thank everyone who contributed toward the success of the workshop: the authors of submitted contributions, the program chairs and the Program Committee for their efforts in reviewing and discussing the submissions under tight time constraints. We are also very grateful to all other ESORICS 2013 organizers whose work ensured a smooth organizational process.

December 2013

Sokratis Katsikas
Isaac Agudo

Organization

Program Chairs

Sokratis Katsikas	University of Piraeus, Greece
Isaac Agudo	University of Malaga, Spain

Publicity Chair

Christopher Dadoyan	University of Piraeus, Greece

Program Committee

Lejla Batina	Radboud University Nijmegen, The Netherlands
Carlos Blanco Bueno	Universidad de Cantabria, Spain
David Chadwick	University of Kent, UK
Sherman S.M. Chow	Chinese University of Hong Kong, Hong Kong
Paolo D'Arco	University di Salerno, Italy
Sabrina De Capitani Di Vimercati	DTI - Universita degli Studi di Milano, Italy
Carmen Fernandez Gago	University of Malaga, Spain
Simone Fischer-Huebner	Karlstad University, Sweden
Sara Foresti	DTI - Universita degli Studi di Milano, Italy
Steven Furnell	University of Plymouth, UK
Dimitris Geneiatakis	University of Piraeus, Greece
Stefanos Gritzalis	University of the Aegean, Greece
Peter Gutmann	University of Auckland, New Zealand
Ravi Jhawar	Università degli Studi di Milano, Italy
Georgios Kambourakis	University of the Aegean, Greece
Dogan Kesdogan	University of Siegen, Germany
Elisavet Konstantinou	University of the Aegean, Greece
Costas Lambrinoudakis	University of Piraeus, Greece
Herbert Leitold	A-SIT, Austria
Dimitris Lekkas	University of the Aegean, Greece
Javier Lopez	University of Malaga, Spain
Fabio Martinelli	IIT-CNR, Italy
Catherine Meadows	NRL, USA
Chris Mitchell	Royal Holloway, University of London, UK
Stig Mjolsnes	Norwegian University of Science and Technology NTNU, Norway

Yi Mu	University of Wollongong, Australia
Svetla Nikova	K.U. Leuven and University of Twente, The Netherlands
Rolf Oppliger	eSECURITY Technologies, Switzerland
Massimiliano Pala	Polytechnic Institute of New York University, USA
Stefano Paraboschi	Università di Bergamo, Italy
Andreas Pashalidis	K.U.Leuven, The Netherlands
Olivier Pereira	Université catholique de Louvain, Belgium
Günther Pernul	Universitt Regensburg, Germany
Nineta Polemi	University of Pireaus, Greece
Sasa Radomirovic	ETH Zürich, Switzerland
Pierangela Samarati	DTI - Universita degli Studi di Milano, Italy
Sean Smith	Dartmouth College, UK
Christos Xenakis	University of Piraeus, Greece

Additional Reviewers

Broser, Christian
Heupel, Marcel
Mavrogiannopoulos, Nikos
Nikov, Ventzi
Peters, Thomas
Reiter, Andreas
Riesner, Moritz

Table of Contents

Partial Model Checking for the Verification and Synthesis of Secure Service Compositions*

Fabio Martinelli and Ilaria Matteucci

IIT-CNR, Pisa, Italy
firstname.lastname@iit.cnr.it

Abstract. Security is one of the main aspects of Web Services composition. In this paper we describe a logical approach based on *partial model checking* technique and *open system analysis* for the *verification* and *synthesis* of secure service orchestrators. Indeed through this framework we are able to specify a system with a possible intruder and verify whether the whole system is secure, *i.e.*, whether the system satisfies a given temporal logic formula that describes a correct behavior (*security property*). Moreover we are able to define an *orchestrator operator* able to orchestrate several services in such a way to guarantee both functional and security requirements.

Keywords: Synthesis of Functional and Secure Processes, Secure Service Composition, Partial Model Checking, Cryptography, Process Algebras, Quantitative Security.

1 Introduction

In the last decades, the research on several aspects of service composition made a great step further. In particular, several frameworks have been developed in order to compose services in order to satisfy requirements and constraints imposed by a user. The Service Oriented Computing (SOC) investigates on new approach for building software applications by composing and configuring existing *services*. Services are software components developed to be re-usable, which expose their definition and which are accessible by third parties. Web Services are the most promising class of services, export their description and are accessible through standard network technologies, *e.g.*, SOAP, WSDL, UDDI, WS-BPEL, WS-Transaction, *etc.*. Web Service Composition combines existing services, available on the web, to provide added-value services featuring higher level functionalities. Every functionality of a service network depends on how the services compose each other. Service composition can be made in two ways, as a *choreography* or through an *orchestration*. Choreography identifies the end-to-end composition between two services by mainly considering cooperation rules, *e.g.*, the sequence of the exchanged messages and their content. Orchestration deals with the composition of multiple services in terms of the business process they generate.

* Work partially supported by the EU project FP7-257930 Aniketos: *Ensuring Trustworthiness and Security in Service Composition* and by the EU project FP7-256980 Nessos: *Network of Excellence on Engineering Secure Future Internet Software Services and Systems*.

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 1–11, 2014.

The pervasiveness of web services increases the necessity for consumers to access and use them in a *secure* way. A service composition is secure whether it satisfies a certain *security property*. A *security property* is a statement that specifies acceptable executions of the system. Indeed, the composition of services presents a lot of challenges in term of security. For instance, services cannot be able to directly communicate one another because they use different cryptographic protocols. It is also possible that different services provide the same functionality but in a different way and one could fit better than an other to the customer functional and security requirements. Furthermore, the distributive nature of web service makes the importance of having some machinary to guarantee security very important. Consumers should require strong guarantees that their security policies are satisfied. Unfortunately, Service Oriented Computing is adverse to most techniques of control and analysis which, usually, require the direct access to either execution or implementation.

In this paper we focus on orchestration. In particular we show our approach for verification and synthesis of secure service composition using a secure and functional orchestrator. Indeed, we consider the research line of verification and synthesis of secure systems using *partial model checking* [1]. Hence, we describe our approach and we also show the tools we have developed in order to verify and generate secure cryptographic orchestrators:

- Verification of security properties by partial model checking through the PaMoChSA tool;
- Synthesis of orchestrators by partial model checking through an extended version of PaMoChSA, the PaMoChSA 2012 version.

The paper is organized as follows. Next section recalls some background notions about process algebra, cryptography and partial model checking function. Section 3 introduces our approach base on the open system paradigm for the specification, analysis, and synthesis of a service composition system. In particular, in this section we focus on the verification framework, while in Section 4 we describe our extension of the framework to automatically synthesize of cryptographic orchestrators that allow services to communicate in both functional and secure way. Furthermore, we present a possible strategy to evaluate and rank different orchestrators according to *cost* of orchestrator execution. Section 5 discusses some related work about the verification and synthesis of secure service composition. Finally Section 6 concludes the paper.

2 Background Notions

In this section we briefly recall some notions about Crypto-CCS [2,3], a variant of the CCS process algebra [4] that allows to deal with cryptography, and partial model checking function [1] that allows to partially evaluate the behaviour of the considered system.

2.1 Crypto-CCS in a Nutshell

Crypto-CCS is a variant of CCS [4], endowed with cryptographic primitives. A model defined in Crypto-CCS consists of a set of sequential agents able to communicate by exchanging messages (*e.g.*, data manipulated by the agents).

$$A \doteq 0 \mid c!m.A \mid c?x.A \mid [m_1 \cdots m_n \vdash_r x]A; A_1$$

where $m_1, \ldots, m_n, m$ are closed messages or variables, x is a variable and c is an element of the set *Ch* of channels. Informally, the Crypto-CCS semantics used in the sequel is: 0 denotes a process that does nothing; $c!m.A$ denotes a message m sent over channel c and then behave as A; $c?x.A$ denotes a message m received over channel c which replaces the variable x and then behave as A; $[m_1 \cdots m_n \vdash_r x]A; A_1$ denotes an *inference* test that a process may use to check whether message m is derivable from premises $m_1, \ldots, m_n$; the continuations in positive and negative cases are A (where m replaces x), or A_1, respectively. Deduction is the message-manipulating construct of the language, responsible for its expressive power. In particular, it allows to model asymmetric encryption. Let y be a key belonging to an asymmetric pair of keys. We denote by y^{-1} the correspondent complementary key. If y is used for encryption, then y^{-1} is used for decryption, and vice versa. Given a set of messages ϕ, then message $m \in \mathcal{D}(\phi)$, the set of deduced messages, if and only if m can be deduced from the rules modelling public key cryptography.

The control part of the language consists of *compound systems*:

$$S \doteq S_1 \parallel S_2 \mid S \backslash L \mid A_\phi$$

Informally, $S_1 \parallel S_2$ denotes the parallel composition of S_1 and S_2, *i.e.*, $S_1 \parallel S_2$ performs an action if either S_1 or S_2 does. A synchronization (or *internal*) action, denoted by τ, is observed whenever S_1 and S_2 can perform two complementary send and receive actions over the same channel; $S \backslash L$ prevents actions whose channels belong to the set L, except for synchronization. A_ϕ is a single sequential agent whose knowledge is described by ϕ.

2.2 Partial Model Checking

Partial model checking is a technique that relies upon compositional methods to provide properties of concurrent systems [1].

The intuitive idea underlying the partial model checking is the following: let φ be a formula expressing a certain consumer's requirement (see [3,5] for some logical languages), then proving that $E \| F$ satisfies φ is equivalent to prove that F satisfies a modified specification $\varphi = \varphi_{//_E}$, where $//_E$ is the partial evaluation function for the parallel composition operator. Hence, the behavior of a component has been partially evaluated and the requirements are changed in order to respect this evaluation.

We give the following main result:

Lemma 1. *Given a process $E \| F$ and an equational specification φ we have:*

$$E \| F \models \varphi \quad \textit{iff} \quad F \models \varphi_{//_E}$$

A lemma similar to the previous one holds for each process algebra operator.

3 Verify Communication Protocols in Service Communication

In order to guarantee security in service composition, we aim to verify and automatically synthesize (see next section) an orchestrator process able to coordinate the communication among several services in a secure and functionally correct way.

Given the sensitive nature of a cryptographic protocol, one can imagine the presence of a hostile adversary trying to interfere with the normal execution of the protocol in order to achieve some advantage. To this aim, hereafter, we assume the Dolev-Yao threat model which has been widely accepted as thread model for cryptographic protocols. This threat model assumes that:

- All communications are visible by the attacker, *i.e.*, an attacker can receive any message transmitted through the network.
- The attacker can alter, forge, replay or drop any message.
- The attacker can reroute messages to another principal.
- The attacker can be a principal or an outsider. This means that an attacker can be a legitimate user of the network and thus in particular he is able to initiate communication with any other principal or to act as a receiver to any principal.

Due to the unpredictable behaviour of the possible attacker, this can be seen as an unspecified component of the system under investigation, *i.e.*, as a *black-box*. Hence, we model as an *open system* following the approach proposed in [6,2,7,3]. A system is *open* if it has some unspecified components. We want to make sure that the system with this unspecified component works properly, *e.g.*, fulfills a certain property. Thus, the intuitive idea underlying the verification of an open system is the following:

An open system satisfies a property if and only if, whatever component is substituted to the unspecified one, the whole system satisfies this property.

Whatever the unspecified term is, it is appealing that the resulting system works properly, *e.g.*, satisfies a consumer's requirement.

According to these premises, using Crypto-CCS we can model the service composition as follows:

$$For\ every\ component\ X \quad S \| X \models \varphi \tag{1}$$

where X stands for the possible attacker, S is the system under examination, consisting of several services composed in parallel through the $\|$ parallel-composition operator, φ is a logic formula expressing the customer requirement. It roughly states that the property φ holds for the system S, regardless of the component (*i.e.*, intruder, malicious user, hostile environment, *etc.*) which may possibly interact with it.

Our aim is to reduce such a verification problem as in Formula (1) to a validity checking problem. To obtain this, we apply the partial model checking techniques.

We have:

$$\forall X \quad S \| X \models \varphi \quad \textit{iff} \quad X \models \varphi_{//_S} \tag{2}$$

In this way we have found the sufficient and necessary condition on X, expressed by a logical formula $\varphi_{//_S}$, so the whole system $S \| X$ satisfies φ.

Several results exist about the decidability of such problems for temporal logic and, for the more interesting properties, like several *safety properties* ("nothing bad happens"), the validity problem of the formula obtained after the partial evaluation may be solved in linear time in the dimension of the formula itself. Another advantage of the partial model checking technique is that it is not necessary to find the most general intruder and prove its attack capabilities.

3.1 PaMoChSA: The Partial Model Checking Security Analyser

The development of the theory has lead to the implementation of a partial model checker namely the Partial Model Checking Security Analyser [8], for short, PaMoChSA, through which it is possible to analyse distributed systems. As usual, only systems with finite computations will be investigated. This is possible since:

1. the operational language used to specify protocols does not allow recursion;
2. the messages are of a fixed structure;
3. a finite number of parties and sessions running the protocol are considered;
4. even if the attacker is allowed to generate fresh messages, their structure is subject to the same constraints mentioned above.

It is worth noticing that, though maintaining the analysis over a finite number of parties and sessions, the absence of attacks over a particular system running the protocol does not guarantee that there are no attacks on larger systems running the same protocol.

The PaMoChSA tool needs the following set of inputs: i) the protocol specification; ii) the security property to be checked; iii) the initial knowledge of the intruder. When developing the theory, the operational language Crypto-CCS has been used for specifying the protocols. The PaMoChSA tool takes as input the protocol description, the secret, *i.e.*, the message that has not to be disclosed to a possible intruder, and the initial knowledge of the possible intruder. The tool gives as output the possible attacks if any, or states the absence of attacks.

4 Synthesis of Functional and Secure Orchestrators

Moving on along this line of research, we wonder what we could do if the verification tool reveals the presence of attacks. In this case, we extend this line of research area based on partial model checking, logic languages and satisfiability, in order to synthesize an orchestrator process able to i) combine several services and provide an unified interface that satisfies a consumer's request and ii) guarantee that the composite service is secure. Hereafter, from a functional perspective, we concentrate on successful service completion, and from a security perspective, we concentrate on the *secrecy* property.

Indeed, let us assume that each service in the composition is not able to communicate with the others for accomplishing the consumer's requirements, *i.e.*, the set of channels over which S_i is able to communicate does not intersect the set of channels over which S_j is able to communicate, for each pair S_i and S_j in S. We may wonder if there exists an orchestrator O that, by communicating with the services in S and assuming any unspecified component X, guarantees that the overall system satisfies the required security property, *i.e.*,

$$\exists O \quad \forall X \quad S \| O \| X \models \varphi$$

Let m_F be a message that denotes the end of a service execution, ϕ_O be the knowledge of the orchestrator, and ϕ_X be the knowledge of the attacker.

The synthesized orchestrator process is consider functional and secure because it is able to:

- **Functional:** combine several services in such a way that m_F falls into the orchestrator's knowledge ϕ_O. This implies that all services have successfully terminated their execution. We consider the formula φ_T for this property.
- **Secure:** guarantee that the composite service is secure by checking that the secret message m does not belong to ϕ_X. We consider the formula φ_{sec} for this property.

Let us consider the process $(S\|O_{\phi_O}\|X_{\phi_X})$. No matter what the behaviour of X is, we require that this process satisfies both functional and security requirements. It is worth noticing that in this case there are two components whose behaviour is unknown: the orchestrator O and the intruder X.

One issue is to *decide* if there exists an orchestrator O such that, for all the possible behaviours of X, after the computation of maximal length $\gamma(max)$, m_F is in the knowledge of O and m is not in the knowledge of X.

$$\exists O_{\phi_O} \forall X_{\phi_X} (S\|O_{\phi_O}\|X_{\phi_X}) \models \varphi_T \wedge \varphi_{sec} \tag{3}$$

An important aspect is how to automatically *synthesize* the orchestrator. We can use partial model checking to simplify Equation 3 by partially evaluating the formula $\varphi_T \wedge \varphi_{sec}$ with respect to the behaviour of S.

Proposition 1. *Let S be a system and O_{ϕ_O} and X_{ϕ_X} two sequential agents, where ϕ_O and ϕ_X are finite sets representing the knowledge of O and X. If m_i, $i = 1, \ldots, n$, are secret messages and m_F is the final one, we have:*

$$(S\|O_{\phi_O}\|X_{\phi_X}) \models \varphi_T \wedge \varphi_{sec}$$
$$\text{iff}$$
$$O_{\phi_O}\|X_{\phi_X} \models (\varphi_T \wedge \varphi_{sec})//ns, S$$

This result identifies the necessary and sufficient conditions that the orchestrator, interacting with every possible X, must satisfy in order to guarantee that the final message m_F is delivered correctly without any disclosure of information to X.

However, the presence of the universal quantifier on X makes the formula $\varphi_T = \forall \gamma(max) : m_F \in K^{\phi_O}_{O,\gamma(max)}$ not satisfiable, since X can always interfere with the normal execution of S getting the overall system stuck, so that the final message m_F is not delivered.

However, still keeping the intuition behind Equation 3, we can weaken the property to the conjunction of the following properties:

A1. When there is not an intruder, the orchestrator always drives the services to correct termination.

A2. When there is an intruder, no matter what actions it takes, it is not able to learn the secret m.

Now we need to determine whether it is possible to determine an orchestrator O satisfying this weaker assumption. Decidability comes from the following proposition.

Proposition 2. *Given a system S, and two finite sets ϕ_O and ϕ_X, it is decidable if* $\exists O_{\phi_O}$ s.t. $\forall X_{\phi_X}$

$$\begin{array}{lll} A1 & (S \| O_{\phi_O}) \setminus L & \models \varphi_T \\ A2 & (S \| O_{\phi_O} \| X_{\phi_X}) \setminus L & \models \varphi_{sec} \end{array}$$

In A1, we are assuming that the attacker X is the empty process, with an empty initial knowledge ϕ_X.

According to Proposition 1, we can apply the partial model checking techniques to A1 and A2 obtaining:

$$\begin{array}{ll} A1' \; O_{\phi_O} & \models (\varphi_T)//_{ns,S} \\ A2' \; (O_{\phi_O} \| X_{\phi}) & \models (\varphi_{sec})//_{ns,S} \end{array}$$

Hence, since the formulas in A1 and A2 are finite, the application of the partial model checking, in conjunction with the usage of some satisfiability procedure allows us to synthesize an orchestrator, whenever it exists.

4.1 PaMoChSA 2012: Tool Description

The tool PaMoChSA2012 [5] is an extension of the original PaMoChSA tool briefly recalled in Section 3.1. It is able to automatically synthesize a functional and secure orchestrator starting from the description of services. The algorithm implements the two formulas of Proposition 2. It can be more intuitively explained as path-finding in a state graph. In principle, the behaviour of an orchestrator is a tree. However, since the system and the orchestrator are assumed to be deterministic, such a tree has an equivalent description in terms of all its paths. The input of PaMoChSA2012 are the same of PaMoChSA plus the initial knowledge of an orchestrator.

A practical way to account for possible attacks is to build the state graph in such a way that additional transitions are present, simulating eavesdropping and manipulation of messages by the intruder. Thus, whenever a service can receive a message from the orchestrator, then it evolves and it can also be instantiated with all messages of the same type that can be deduced from the knowledge of the intruder K_X. Likewise, whenever the service can send a message to the orchestrator, the knowledge K'_O of the orchestrator can also be augmented with all the messages of the same type that can be deduced from K_X. This machinery implements the ability of X to interfere with communications between the orchestrator and the system.

Finally, the knowledge of the intruder is always augmented with the messages that are exchanged between the orchestrator and the system, unless the used channel is in H. Rationale is that the intruder can eavesdrop such communications in order to acquire new information.

4.2 A Selection Strategies of the Best Orchestrator

The proposed approach is able to synthesize functional and secure orchestrators able to orchestrate a given set of services. We present several strategies for ordering the set of orchestrator processes obtained through the PaMoChSA2012 tool by evaluating the orchestrators according to some aspects that can me modelled as a metric. In particular, several aspects, as cost, QoS, and also the degree of security, can be formalised using semirings, as done in [9].

Definition 1. *A* semiring $\mathbb{K} = (K, +, *, \mathbf{0}, \mathbf{1})$ *consists of a set K with two binary operations $+, *$, and two constants $\mathbf{0}, \mathbf{1}$, such that $+$ is associative, with neutral element $\mathbf{0}$; $*$ is associative, with neutral and absorbing elements $\mathbf{1}, \mathbf{0}$; $*$ distributes over $+$.*

Every semiring is endowed with a partial order that intuitively indicates a notion of preference. This allow us to order the synthesized orchestrators they are evaluated according to one of the chosen measures. Let us consider the semiring of *cost*, $\mathbb{K}_C = \langle \mathbb{R}_0^+, min, +, +\infty, 0 \rangle$. Using this semiring we are able to evaluate the cost of each orchestration strategy by associating a cost to each action the orchestrator performs.

As we are dealing with *run-time* orchestration, we work with *traces*, or paths, of the orchestrator processes.

A *path* is a sequence $(a_1, k_1) \cdots (a_n, k_n)$, and we call $\mathcal{T}(A)$ the set of paths rooted in A. Given a path $(a_1, k_1) \cdots (a_n, k_n)$, we define its *label* $l(t) = a_1 \cdots a_n$, and its *run weight* $|t| = k_1 * \ldots * k_n \in K$. Finally, the *valuation* of a process A is given by $[\![A]\!] = \sum_{\{t \in \mathcal{T}(A)\}} |t|$.

Hence, we are able to compare different orchestrator processes.

Definition 2. *Given a process S, an orchestrator O_2 is* better *than an orchestrator O_1 with respect to S, if and only if $[\![O_1 \| S]\!] \subseteq [\![O_2 \| S]\!]$.*

This definition does not directly depend on the semiring used to quantify the controlled target, and it is therefore possible to use the same definition to say that an orchestrator is better than another one with respect to any other measure. Note that since each individual trace can be represented as a target, $O_1 \subseteq O_2$ implies that the valuation of O_1 should be lower than that of O_2 for every possible trace.

In some cases, orchestrators can be incomparable. However, other dimensions can easily be included within our framework, with the intuition that the more accurate is the quantification of the composed system, the more informed is the security designer to choose an orchestrator.

5 Related Work

Several works deal with a possible modeling of orchestrators by process algebras, see *e.g.*, [10,11,12,13,14] or by automata [15]. In [16,17] the authors have developed a static approach to deal with the composition of web services problem by the usage of *plans*. By the way, only some of these take into account also security aspects in the service composition procedure. In particular they use a distributed, enriched λ-calculus for describing networks of services. Both, services and their clients, can protect themselves, by imposing security constraints on each other's behavior. Then, service interaction results in a *call-by-property* mechanism (see [18]), that matches the client requests with services. Our approach treats the problem of the automatic composition of services by also considering cryptographic primitives. Indeed, our approach permits us to synthesize orchestrators that are able to encrypt and decrypt messages in order to guarantee also secrecy and privacy properties.

In [23] the authors introduce COWS, *calculus for orchestration of web services*, as a new foundational language for service oriented computing. In order to facilitate the use of model-checking techniques to business analysts, the authors of [24] created a model-checking plugin for SAP NetWeaver Business Process Management. This plugin support the verification of secrecy properties with a push of a button and the subsequent visualization of possible attack traces. However, since this plugin is intended as a design tool, the designer is left with the task to solve possible flaws in the business process. Our approach, on the other hand, automatizes the generation of secure orchestrators which are guaranteed to preserve the given secrecy properties.

There are some papers proposing compositional approaches to the synthesis of controllers, able to dynamically enhance security, depending on some runtime behaviour of a possible attacker, *e.g.*, [25,26]. The current work extends the existing research line on the synthesis of secure controller programs [25] with the introduction of cryptographic primitives. Also, it tries to simplify the approach in [27] for the synthesis of deadlock-free orchestrators that are compliant with security adaptation contracts [28]. Compared to [27], this new approach loses the ability to specify fine-grained constrains in the desired orchestration but, on the other hand, there is no need to design and adaptation contract.

Similarly, our approach to synthesis differs from the one in [29], where automatic composition of services under security policies is investigated. Work in [29] uses the AVISPA tool [30] and acts in two stages: first, it derives a protocol allowing composition of some services; then, some desired security properties are implemented. The latter step uses the functionality of AVISPA and, for the former step, the desired composition is turn into a security property, so that AVISPA itself can be used to derive an "attacker"which actually is the orchestrator. The AVANTASSAR tool [31] extends the AVISPA tool. Indeed, the AVANTSSAR Platform is an integrated toolset for the formal specification and automated validation of trust and security of service-oriented architectures and other applications in the Internet of Services.

In [32], Li et al. present an approach for securing distributed adaptation. A plan is synthesized and executed, allowing the different parties to apply a set of data transformations in a distributed fashion. In particular, the authors synthesize "security boxes"that wrap services, providing them with the appropriate cryptographic capabilities. Security boxes are pre-designed, but interchangeable at run time. In our case the orchestrator is synthesized at run time and is able to cryptographically arrange secure service composition.

6 Conclusion

In this paper we present our framework based on partial model checking for guaranteeing security in web service composition. In particular, we exploit cryptographic protocols analysis for checking that the communication among different services happens in a secure way. Furthermore, we extend the same framework for synthesizing an orchestrator process able to manage the communication among services by using also cryptographic primitives. We also enrich this approach by presenting a possible strategy to evaluate and ranking different orchestrators processes in order to chose the best one for accomplishing the customer request.

References

1. Andersen, H.R.: Partial model checking. In: LICS, p. 398. IEEE (1995)
2. Martinelli, F.: Languages for description and analysis of authentication protocols. In: Proceedings of 6th Italian Conference on Theoretical Computer Science, pp. 304–315 (1998)
3. Martinelli, F.: Analysis of security protocols as open systems. TCS 290(1), 1057–1106 (2003)
4. Milner, R.: Communication and Concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)
5. Ciancia, V., Martin, J.A., Martinelli, F., Matteucci, I., Petrocchi, M., Pimentel, E.: A tool for the synthesis of cryptographic orchestrators. In: Model Driven Security Workshop, MDSEC. ACM (2012)
6. Martinelli, F.: Formal Methods for the Analysis of Open Systems with Applications to Security Properties. PhD thesis, University of Siena (1998)
7. Marchignoli, D., Martinelli, F.: Automatic verification of cryptographic protocols through compositional analysis techniques. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 148–162. Springer, Heidelberg (1999)
8. Martinelli, F., Petrocchi, M., Vaccarelli, A.: Formal analysis of some secure procedures for certificate delivery. Softw. Test., Verif. Reliab. 16(1), 33–59 (2006)
9. Ciancia, V., Martinelli, F., Matteucci, I., Morisset, C.: Quantitative evaluation of enforcement strategies. In: Proceedings of FPS 2013. LNCS, vol. 8352 (2013)
10. Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Reasoning about interaction protocols for web service composition. Electr. Notes Theor. Comput. Sci. 105, 21–36 (2004)
11. Bao, L., Zhang, W., Zhang, X.: Describing and Verifying Web Service Using CCS. In: PDCAT
12. Cámara, J., Canal, C., Cubo, J., Vallecillo, A.: Formalizing wsbpel business processes using process algebra. Electr. Notes Theor. Comput. Sci. 154(1), 159–173 (2006)
13. Ferrara, A.: Web services: a process algebra approach. In: Aiello, M., Aoyama, M., Curbera, F., Papazoglou, M.P. (eds.) ICSOC, pp. 242–251. ACM (2004)
14. Salaun, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2004), p. 43. IEEE Computer Society, Washington, DC (2004)
15. Reisig, W.: Modeling- and analysis techniques for web services and business processes. In: Steffen, M., Zavattaro, G. (eds.) FMOODS 2005. LNCS, vol. 3535, pp. 243–258. Springer, Heidelberg (2005)
16. Bartoletti, M., Degano, P., Ferrari, G.L.: Plans for service composition. In: Workshop on Issues in the Theory of Security, WITS (2006)
17. Bartoletti, M., Degano, P., Ferrari, G.L.: Types and effects for secure service orchestration. In: Proc. 19th Computer Security Foundations Workshop, CSFW (2006)
18. Bartoletti, M., Degano, P., Ferrari, G.L.: Security issues in service composition. In: Gorrieri, R., Wehrheim, H. (eds.) FMOODS 2006. LNCS, vol. 4037, pp. 1–16. Springer, Heidelberg (2006)

19. Pistore, M., Roberti, P., Traverso, P.: Process-level composition of executable web services: “On-the-fly” versus “Once-for-all” composition. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 62–77. Springer, Heidelberg (2005)
20. Pistore, M., Traverso, P., Bertoli, P.: Automated composition of web services by planning in asynchronous domains. In: Biundo, S., Myers, K.L., Rajan, K. (eds.) ICAPS, pp. 2–11. AAAI (2005)
21. Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., Zavattaro, G.: Choreography and orchestration: A synergic approach for system design. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 228–240. Springer, Heidelberg (2005)
22. Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: Securebpmn: modeling and enforcing access control requirements in business processes. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT 2012, pp. 123–126. ACM, New York (2012)
23. Lapadula, A., Pugliese, R., Tiezzi, F.: A calculus for orchestration of web services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 33–47. Springer, Heidelberg (2007)
24. Arsac, W., Compagna, L., Pellegrino, G., Ponta, S.E.: Security validation of business processes via model-checking. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 29–42. Springer, Heidelberg (2011)
25. Martinelli, F., Matteucci, I.: A framework for automatic generation of security controller. STVR (2010)
26. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 229–242. Springer, Heidelberg (1995)
27. Martín, J.A., Martinelli, F., Pimentel, E.: Synthesis of secure adaptors. J. Log. Algebr. Program. 81(2), 99–126 (2012)
28. Martín, J.A., Pimentel, E.: Contracts for security adaptation. J. Log. Algebr. Program. 80(3-5), 154–179 (2011)
29. Chevalier, Y., Mekki, M.A., Rusinowitch, M.: Automatic composition of services with security policies. In: IEEE SERVICES 2008 - Part I, pp. 529–537 (2008)
30. Viganò, L.: Automated security protocol analysis with the AVISPA tool. ENTCS 155, 69–86 (2006)
31. Armando, A., et al.: The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 267–282. Springer, Heidelberg (2012)
32. Li, J., Yarvis, M., Reiher, P.: Securing distributed adaptation. Computer Networks 38(3) (2002)

Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures

Christina Brzuska[1,⋆], Henrich C. Pöhls[2,4,⋆⋆], and Kai Samelin[3,4,⋆⋆⋆]

[1] Tel Aviv University, Israel
[2] Chair of IT-Security
[3] Chair of Computer Networks and Computer Communication
[4] Institute of IT-Security and Security Law (ISL), University of Passau, Germany
brzuska@post.tau.ac.il, {hp,ks}@sec.uni-passau.de

Abstract. Sanitizable signatures allow for controlled modification of signed data. The essential security requirements are accountability, privacy and unlinkability. Unlinkability is a strong notion of privacy. Namely, it makes it hard to link two sanitized messages that were derived from the same message-signature pair. In this work, we strengthen the standard unlinkability definition by *Brzuska* et al. at PKC '10, making it robust against malicious or buggy signers. While state-of-the art schemes deploy costly group signatures to achieve unlinkability, our construction uses standard digital signatures, which makes them compatible with existing infrastructure.

We construct a sanitizable signature scheme that satisfies the strong notion of perfect unlinkability and, simultaneously, achieves the strongest notion of accountability, i.e., non-interactive public accountability. Our construction is not only legally compliant, but also highly efficient, as the measurements of our reference implementation show. Finally, we revisit the security model by *Canard* et al. and correct a small flaw in their security definition given at AfricaCrypt '12.

1 Introduction

Sanitizable signature schemes (SanSigs), introduced by *Ateniese* et al. [3], enable a designated party, the sanitizer, to alter a signed document in a controlled way. The sanitizer (holding its own sanitizer secret) can generate a new, yet valid, signature for the modified document without interacting with the signer of the

[⋆] Was supported by the Israel Science Foundation (grant 1076/11 and 1155/11), the Israel Ministry of Science and Technology (grant 3-9094), and the German-Israeli Foundation for Scientific Research and Development (grant 1152/2011).

[⋆⋆] Is funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUeIT project.

[⋆⋆⋆] The research leading to these results was supported by "Regionale Wettbewerbsfähigkeit und Beschäftigung", Bayern, 2007-2013 (EFRE) as part of the SECBIT project (http://www.secbit.de) and the European Community's Seventh Framework Programme through the EINS Network of Excellence (grant agreement no. [288021]).

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 12–30, 2014.

original document. In particular, let the message m consists of ℓ blocks, i.e., $m = (m[1], \ldots, m[\ell])$, where $\ell \in \mathbb{N}$ and $m[i] \in \{0,1\}^*$. Then, the sanitizer is only able to modify those blocks $m[i]$ that the signer defined as admissible. Sanitization thus yields a new message-signature pair (m', σ'), where σ' is a valid signature for m' under the signer's public key pk_{sig} and m' is equal to m on all *non*-admissible blocks.

Motivation. Malleable signatures of this kind seem to bear an inherent risk: a semi-trusted party is allowed to change signed data and thus a signer gives up control over the statements that are produced in its name. However, when carefully implemented, delegation of signing rights turns out to be very useful for a variety of application scenarios, ranging from sanitizing medical records to secure routing and blank signatures [3,9,22,23]. Another application scenario is access control for databases. In any larger company, but in particular in banks and hospitals, access policies are inherent in day-to-day operations [39]. Compliance rules in banks actually enforce the separation of different sectors, and likewise, hospitals host large databases of sensitive data that must not be accessed by anybody. The reception desk personnel in a hospital, for example, must not be able to access medical data of a patient, while the accountant of the hospital, in turn, must not learn personally identifying details of the patient. On the other hand, integrity of the database is crucial for both, hospitals and banks, and besides appropriate read-and-write policies [39], one might aim for the cryptographic protection of, say, patient records and have them digitally signed by the treating personnel. The accountant and the receptionist then both access different parts of an authenticity protected record. Hence, some entries in the signed record need to be sanitized while keeping a valid signature over the rest of the record. Whereas standard signatures do not allow for such modifications, sanitizable signatures enable to implement privacy-friendly access and integrity verifiability simultaneously. In particular, using sanitizable signatures, the database can operate without repeated interaction with the signer (the medical personnel). In a bank, interaction with the signer might even be disallowed, e.g., due to money laundering policies. To sum up, we aim for signatures that allow for controlled modification of different parts of a signed document without interaction with the original signer.

Sanitizable signatures usually strive for strong privacy guarantees to hide the removed sensitive information. The strongest notion of privacy is unlinkability, which we review next. In the hospital database example, we derive two different sanitized documents from the original patient's record, as we removed different parts from the same signed patient record to create a version for the accountant and another one for the reception desk personnel. A secure solution must prevent inferring information about the original record by combining the two sanitized documents, as this would violate the patient's privacy concerns as well as data-protection regulations such as HIPAA [15]. Thus, in such application domains [10], it is important for sanitizable signatures schemes to achieve unlinkability.

A digital signature on a document usually provides a legal value of evidence [34]. For example, in a hospital, the medical personnel sign their entries in the database and can be held accountable for those later, i.e., the signature allows for identifying the doctor or nurse that generated an entry. For SanSigs there are two different options for accountability: interactive accountability involves the signer and allows an authority to trace back the origin of a message-signature pair, while the scheme itself might be transparent, i.e., the origin of a signature is hidden from third-parties[1]. The alternative is a non-interactive public form of accountability [11] where third-parties can identify immediately whether the medical record was sanitized or not without interaction with the signer. As observed by *Pöhls* and *Höhne* [34], current legislation only attributes a high value of evidence to sanitizable signature if *any* subsequent change can be detected, which is incompatible with the property of transparency. Formally, *Brzuska* et al. show how to achieve public accountability in the absence of transparency, as the two are mutually exclusive [11]. In turn, privacy and a public, i.e., non-interactive, form of accountability can be achieved simultaneously, as *Brzuska* et al. [11] show. We continue this line of practical research and strengthen the privacy by achieving perfect unlinkability with standard primitives.

Challenges and Contributions. In a nutshell, our scheme addresses the following challenges: (1) Strong privacy guarantees, namely perfect unlinkability and perfect privacy, (2) a high legal value of evidence through non-interactive public accountability, (3) compatibility with existing public key infrastructures (PKI) for standard signatures, (4) performance restrictions for economic applicability. From a practitioner's point of view, having a legally recognizable *and* extremely efficient scheme is essential for deployment of a signature scheme [35,36]. Our scheme is based on the ideas given in [9,11]. It extends their work in the aforementioned points. Essentially, we achieve stronger privacy and accountability properties with simpler building blocks. Moreover, we are also able to consider unlinkability and strengthen the original definition from *Brzuska* et al. from PKC '10 [10] to be more robust against malicious or buggy signers, as well as corruption of the signer's key. Interestingly, the new definition turns out to be more compact than the original one, as the oracles that use the signer's secret key can now be simulated by the adversary. Hence, the Sign and the Proof oracle of [10] are not required anymore. We also correct a small flaw in the security definition of unlinkability in the multi-sanitizer setting given by *Canard* et al. at AfricaCrypt '12 [13]. There, the LoRSanit-oracle does not check whether *both* inputs are valid message-signature pairs and whether the requested modification is admissible. Thus, as we show, the original definition is not achievable. Finally, we show how to switch on-the-fly between unlinkability and linkability.

Techniques. The first sanitizable signature scheme [3] was based on chameleon hashes [28] that were applied per each admissible block. However, their construction allowed for mix-and-match attacks. Later schemes had to find a collision

[1] Third-parties here meaning other than signer or sanitizer.

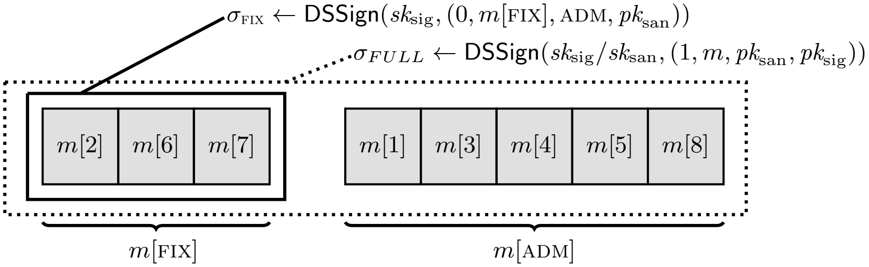
$\sigma_{\text{FIX}} \leftarrow \mathsf{DSSign}(sk_{\text{sig}}, (0, m[\text{FIX}], \text{ADM}, pk_{\text{san}}))$
$\sigma_{FULL} \leftarrow \mathsf{DSSign}(sk_{\text{sig}}/sk_{\text{san}}, (1, m, pk_{\text{san}}, pk_{\text{sig}}))$
$m[2]$
$m[6]$
$m[7]$
$m[1]$
$m[3]$
$m[4]$
$m[5]$
$m[8]$
$m[\text{FIX}]$
$m[\text{ADM}]$

Fig. 1. Blocks 2,6,7 of m are fixed and together with pk_{san} and ADM signed by signer (σ_{FIX}). The complete message m is signed by either the signer or the sanitizer (σ_{FULL}).

on each admissible blocks, not only those modified [8]. Thereby, the sanitizing process was linear in the number of admissible blocks. Some later constructions [10,9,11] were based on a different paradigm. The idea is to use two signatures (see Fig. 1); one to sign the fix part of m, i.e., $m[\text{FIX}]$, we sometimes call this the "inner signature", and another one to sign the admissible parts, i.e., $m[\text{ADM}]$, together with the fixed parts, often called the "outer signature". The inner signature is produced by the signer of the signature scheme, while the outer signature can be produced by either one, the signer or the sanitizer. Using different signature types as inner and outer signature yields different properties of the sanitizable signature scheme. For instance, *Brzuska* et al. [10] use a group signature for the outer signature. The anonymity of the group signature makes signatures of the signer and the sanitizer indistinguishable, and the non-frameability/traceability property of the group signature scheme assures an interactive form of accountability. In turn, in [9] and [11], the authors use standard signature schemes also for the outer signature. The scheme becomes very efficient; it is not transparent anymore, but it still enjoys privacy [9,11] and a non-interactive public form of accountability [11], thus complying with legal standards. As transparency is sometimes seen as a stronger notion of privacy, one might feel that one has to compromise, as one cannot obtain a strong notion of accountability and a strong notion of privacy simultaneously. We show that this is actually not the case. As the inner signature scheme, we use a deterministic signature scheme and prove that the scheme satisfies both, a public, non-interactive version of accountability and a statistical notion of unlinkability, the strongest notion of privacy. At the same time, we maintain high efficiency and compatibility with existing public-key infrastructure, as our scheme only requires a constant number of standard building block operations. Our construction is even less complex than the ones given in [41], as we do not deploy labels. This makes our scheme applicable for use on Smart Cards [35], embedded devices like routers, or other devices which do not have as much processing power. All of these requirements are of paramount importance to make SanSigs used in practice.

State-of-the-Art and Related Work. Malleable signatures scheme have gained a lot of attention in the past few years. They were studied in several flavors, for example, redactable signature schemes [16,25,32,33,40], sanitizable signatures with several extensions [9,12,18,22,27,30,36,42] and combinations of both approaches [24]. Moreover, the integrity protection of structured data, such as in [7,29,37,38], has equally been studied in the recent past. All schemes aim at the same goal: allowing for controlled modification of signed data to preserve privacy, while retaining authentication of origin and integrity protection against uncontrolled, i.e., unauthorized, modifications. In this paper, we focus on sanitizable signatures, as introduced by *Ateniese* et al. [3]. They also introduced the aforementioned security properties for sanitizable signatures, namely privacy, immutability, accountability, transparency and unforgeability. These were later formalized and extended by *Brzuska* et al. [8,10]. Their framework has been extended to multi-sanitizer environments by *Canard* et al. [13]. We work within the original framework for a single signer and a single sanitizer and show in Sect. 4 how to modify our scheme to the multi-user setting from [13].

The stronger notion of *statistical* unlinkability for *redactable* signature schemes was introduced by *Ahn* et al. at TCC '12 [1], and even stronger notions have been discussed recently in [4,5,19]. Neither of these notions has been considered in the context of sanitizable signatures yet, and we address this gap. The schemes for *quoting* substrings [1,4,5] are tailored towards achieving both, statistical transparency and statistical unlinkability. This ambitious goal comes at the price of weakening the unforgeability property to selective unforgeability in the case of [1]. Additionally, none of the mentioned schemes is accountable. By trading in transparency, our construction does not only achieve stronger notions of unlinkability and accountability, but also the standard adaptive unforgeability notion instead of selective unforgeability. In the context of sanitizable signatures, the notion of unlinkability captures that two sanitized *messages* cannot be linked to having the same original *message-signature pair.* For group signatures [17], in turn, the unlinkability definition corresponds to the anonymity of the signer, which is usually called transparency in the context of sanitizable signatures. The different nomenclature is maybe best explained by the fact that discussions in the area of malleable signatures are message-centered, while the way of thinking in the area of group signatures is more signer-centered—after all, the word "group" refers to a group of signers, not to a group of messages. To avoid confusion due to the historical evolution of the properties' names in the two areas, we stress that the present paper uses the nomenclature as introduced in [10]. Another related concept are proxy signatures [31]. However, they allow for delegating signing rights entirely, while sanitizable signatures allow for *altering* a specific signed message.

2 Security Models

2.1 Syntax and Notation

For a message $m = (m[1], \ldots, m[\ell])$, we call $m[i] \in \{0,1\}^*$ a *block*, while the special symbol "," $\notin \{0,1\}^*$ denotes a uniquely reversible concatenation of strings. The special symbol $\perp \ \notin \{0,1\}^*$ denotes an error or an exception. The following nomenclature is adapted from *Brzuska* et al. [8], who address a setting of single signers and sanitizers. We also elaborate in Sect. 4 on how to extend our construction for multi-sanitizer environments as described by *Canard* et al. [13].

Definition 1 (Sanitizable Signature Scheme). *Any SanSig consists of at least seven efficient, i.e., $\mathcal{PPT}$ algorithms. In particular, let $\mathsf{SanSig} := (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$, such that:*

Key Generation. *There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys, a private key and the corresponding public key, based on the security parameter λ:*

$$(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda) \quad (\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$$

Signing. *The Sign algorithm takes as input the security parameter λ, a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$, the secret key sk_{sig} of the signer, the public key pk_{san} of the sanitizer, as well as a description ADM of the admissibly modifiable blocks. In detail, ADM contains a set of indices of the modifiable blocks and the overall number ℓ of blocks in m, to guard against length-altering attacks. The Sign algorithm outputs the message m and a signature σ (or $\perp$, indicating an error):*

$$(m, \sigma) \leftarrow \mathsf{Sign}(1^\lambda, m, \mathrm{sk}_{sig}, \mathrm{pk}_{san}, \mathrm{ADM})$$

Sanitizing. *The algorithm Sanit takes a message $m = (m[1], \ldots, m[\ell])$, where $m[i] \in \{0,1\}^*$, a signature σ, the security parameter λ, the public key pk_{sig} of the signer and the secret key sk_{san} of the sanitizer. It modifies the message m according to the modification instruction MOD, which contains pairs $(i, m[i]')$ describing the index i and the blocks new value $m[i]'$. We use the shorthand notation of $m' \leftarrow \mathrm{MOD}(m)$ to denote that the modification instructions were successfully applied to create the modified m' from m. Note, MOD could also be empty, i.e., $m' = m$ is generally possible. Sanit calculates a new signature σ' for the modified message $m' \leftarrow \mathrm{MOD}(m)$. The Sanit algorithm outputs m' and σ' (or possibly $\perp$ in case of an error).*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(1^\lambda, m, \mathrm{MOD}, \sigma, \mathrm{pk}_{sig}, \mathrm{sk}_{san})$$

Verification. *The algorithm Verify outputs a decision $d \in \{0,1\}$ verifying the correctness of a signature σ for a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$ with respect to the public keys pk_{sig} and pk_{san} and the security parameter λ:*

$$d \leftarrow \mathsf{Verify}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san})$$

Proof. *The algorithm* Proof *takes as input the security parameter, the secret signing key* sk_{sig}*, a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a signature* σ *as well a set of (polynomially bounded) additional message-signature pairs* $\{(m_i, \sigma_i) \mid i \in \mathbb{N}\}$ *and the public key* pk_{san}*. The* Proof *algorithm outputs a string* $\pi \in \{0,1\}^*$ *(or* $\bot$*, indicating an error):*

$$\pi \leftarrow \mathsf{Proof}(1^\lambda, \mathrm{sk}_{sig}, m, \sigma, \{(m_i, \sigma_i) \mid i \in \mathbb{N}\}, \mathrm{pk}_{san})$$

Judge. *The algorithm* Judge *takes as input the security parameter, a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a valid signature* σ*, the public keys of the parties and a proof* π*. The* Judge *algorithm outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}, \bot\}$ *indicating whether the message-signature pair has been created by the signer or the sanitizer (or* $\bot$*, indicating an error):*

$$d \leftarrow \mathsf{Judge}(1^\lambda, m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}, \pi)$$

We require that for every SanSig the usual correctness properties hold. That is, for every genuinely created message-signature pair created by the Sign algorithm, the Verify algorithm outputs 1 with overwhelming probability. From every message-signature pair with a positive verification result, the Sanit algorithm produces again a message-signature pair for which the Verify algorithm outputs 1 with overwhelming probability. The latter is recursive, i.e., sanitized message-signature pairs can be sanitized again. For every genuinely generated value of the proof π, the Judge algorithm outputs the correct party, i.e., Sig or San, with overwhelming probability. For more details on the definition of correctness, refer to [8]. In case of a SanSig with non-interactive public accountability, the Proof algorithm always returns $\bot$, and Judge correctly decides upon such an empty proof ($\pi = \bot$) with overwhelming probability.

We write $m[\textsc{adm}]$ for the uniquely reversible concatenation of the *admissible* blocks and $m[\textsc{fix}]$ for the uniquely reversible concatenation of the *fixed* blocks in the order of appearance in m. Note, we do not attach any labels to the blocks as done in [41]. We require the public key to be efficiently derivable from its corresponding secret key. Additionally, we require that ADM is always correctly recoverable from any valid signature σ, which accounts for the findings by [21].

2.2 Security of Sanitizable Signatures

We present an extended security model based on [8]. It covers the basic ideas and features of a SanSig.

Definition 2 (Privacy). *A sanitizable signature scheme* SanSig *is private, if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ *given in Fig. 2 returns* 1*, is negligibly close to* $\frac{1}{2}$ *(as a function of* λ*). Here, the adversary must be able to decide which input was chosen by the* LoRSanit*. The oracle signs and sanitizes the data itself.*

```
$\textit{Experiment}\ \mathsf{Privacy}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\lambda})$
  $(pk_{\mathrm{san}}, pk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\lambda})$
  $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\mathrm{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\mathrm{san}})}_{\mathsf{Proof}(sk_{\mathrm{sig}},\cdots),\mathsf{LoRSanit}(\cdots,sk_{\mathrm{sig}},sk_{\mathrm{san}},b)}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    where oracle $\mathsf{LoRSanit}$ on input of:
    $m_{0,i}, \textsc{mod}_{0,i}, m_{1,i}, \textsc{mod}_{1,i}, \textsc{adm}_i$
    if $\textsc{mod}_{0,i} \not\subseteq \textsc{adm}_i$, return $\perp$
    if $\textsc{mod}_{1,i} \not\subseteq \textsc{adm}_i$, return $\perp$
    if $\textsc{mod}_{0,i}(m_{0,i}) \neq \textsc{mod}_{1,i}(m_{1,i})$, return $\perp$
    let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(m_{b,i}, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \textsc{adm}_i)$
    return $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(m_i, \textsc{mod}_{b,i}, \sigma, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
  return 1, if $a = b$
```

Fig. 2. Privacy

```
$\textit{Experiment}\ \mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\lambda})$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\lambda})$
  $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\mathrm{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\mathrm{san}})}_{\mathsf{Proof}(sk_{\mathrm{sig}},\cdots),\mathsf{Sanit/Sign}(\cdots,sk_{\mathrm{sig}},sk_{\mathrm{san}},b)}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    where oracle $\mathsf{Sanit/Sign}$ for input $m_i, \textsc{mod}_i, \textsc{adm}_i$
    let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(1^{\lambda}, m_i, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \textsc{adm}_i)$,
    compute $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(m_i, \textsc{mod}_i, \sigma_i, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
    if $b = 1$:
      compute $(m'_i, \sigma'_i) \leftarrow \mathsf{Sign}(m'_i, sk_{\mathrm{sig}}, pk_{\mathrm{san}}, \textsc{adm}_i)$
    finally return $(m'_i, \sigma'_i)$.
  return 1 if $a = b$ and $\mathcal{A}$ has not queried
  any $(m_i, \sigma_i)$ output by $\mathsf{Sanit/Sign}$ to $\mathsf{Proof}$.
```

Fig. 3. Transparency

```
$\textit{Experiment}\ \mathsf{Unlinkability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^{\lambda})$
  $(pk_{\mathrm{san}}, pk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\lambda})$
  $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\mathrm{sig}},\cdots),\mathsf{Sanit}(\cdots,sk_{\mathrm{san}})}_{\mathsf{Proof}(sk_{\mathrm{sig}},\cdots)\mathsf{LoRSanit}(\cdots,sk_{\mathrm{san}},sk_{\mathrm{sig}},b)}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    where oracle $\mathsf{LoRSanit}$ on input of:
    $m_{0,i}, \textsc{mod}_{0,i}, \sigma_{0,i}, m_{1,i}, \textsc{mod}_{1,i}, \sigma_{1,i}$
    //$\textsc{adm}$ needs to be recoverable from all $\sigma$
    if $\textsc{adm}_{0,i} \neq \textsc{adm}_{1,i}$, return $\perp$
    if $\textsc{mod}_{0,i} \not\subseteq \textsc{adm}_{0,i}$, return $\perp$
    if $\textsc{mod}_{1,i} \not\subseteq \textsc{adm}_{1,i}$, return $\perp$
    if $\textsc{mod}_{0,i}(m_{0,i}) \neq \textsc{mod}_{1,i}(m_{1,i})$, return $\perp$
    if $\mathsf{Verify}(1^{\lambda}, m_{0,i}, \sigma_{0,i}, pk_{\mathrm{sig}}, pk_{\mathrm{san}}) \neq 1$ or
    $\mathsf{Verify}(1^{\lambda}, m_{1,i}, \sigma_{1,i}, pk_{\mathrm{sig}}, pk_{\mathrm{san}}) \neq 1$, return $\perp$
    return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_{b,i}, \textsc{mod}_{b,i}, \sigma_{b,i}, pk_{\mathrm{sig}}, sk_{\mathrm{san}})$
  return 1, if $a = b$
```

Fig. 4. Unlinkability by *Brzuska* et al.

```
$\textit{Experiment}\ \mathsf{UnlinkabilityEx}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda, n, m)$
  $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^{\lambda}, n, m)$
  $b \leftarrow \{0, 1\}$
  $(m_0, \textsc{mod}_0, \sigma_0, m_1, \textsc{mod}_1, \sigma_1, j_0, j_1, st) \leftarrow \mathcal{A}_{ch}^{(*)}(\mathsf{PK})$
  let $(m', \sigma') \leftarrow \mathsf{Sanit}(m_b, \sigma_b, \mathsf{SK}[j_b], \textsc{mod}_b, \mathsf{PK})$
  let $(\mathsf{ORI}, I_{\mathrm{ORI}}, \pi_{ORI}) \leftarrow \mathsf{FindOri}(m', \sigma'_b, \mathsf{osk}_{\mathrm{ORI}}, \mathsf{PK})$
  if $I_{\mathrm{ORI}} = 0$ or $(I_{\mathrm{ORI}} = (\mathtt{Sig}, i_{\mathrm{ORI}})$ and $i_{\mathrm{ORI}} \in \mathcal{CU})$ or
  $j_0 \in \mathcal{CU}$ or $j_1 \in \mathcal{CU}$ or
  $\mathsf{Judge}(m', \sigma'_b, (\mathsf{ORI}, i_{\mathrm{ORI}}, \pi_{\mathrm{ORI}}), \mathsf{PK}) = 0$, return $\perp$
  let $b^* \leftarrow \mathcal{A}_{gu}^{(*)}(m', \sigma'_b, st)$
  if $(m', \sigma')$ was queried to $\mathsf{SigOpen}$, return $\perp$, else $b^*$
```

Fig. 5. Unlinkability by *Canard* et al.

```
$\textit{Experiment}\ \mathsf{SUnlinkability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^{\lambda})$
  $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdots,sk_{\mathrm{san}}),\mathsf{LoRSanit}(\cdots,sk_{\mathrm{san}},b)}(pk_{\mathrm{san}})$
    where oracle $\mathsf{LoRSanit}$ on input of:
    $m_{0,i}, \textsc{mod}_{0,i}, \sigma_{0,i}, m_{1,i}, \textsc{mod}_{1,i}, \sigma_{1,i}, pk_{\mathrm{sig},i}$
    //$\textsc{adm}$ needs to be recoverable from all $\sigma$
    if $\textsc{adm}_{0,i} \neq \textsc{adm}_{1,i}$, return $\perp$
    if $\textsc{mod}_{0,i} \not\subseteq \textsc{adm}_{0,i}$, or $\textsc{mod}_{1,i} \not\subseteq \textsc{adm}_{1,i}$, or $\textsc{mod}_{0,i}(m_{0,i}) \neq \textsc{mod}_{1,i}(m_{1,i})$, return $\perp$
    if $\mathsf{Verify}(1^{\lambda}, m_{0,i}, \sigma_{0,i}, pk_{\mathrm{sig},i}, pk_{\mathrm{san}}) \neq 1$ or $\mathsf{Verify}(1^{\lambda}, m_{1,i}, \sigma_{1,i}, pk_{\mathrm{sig},i}, pk_{\mathrm{san}}) \neq 1$, return $\perp$
    return $(m', \sigma') \leftarrow \mathsf{Sanit}(m_{b,i}, \textsc{mod}_{b,i}, \sigma_{b,i}, pk_{\mathrm{sig},i}, sk_{\mathrm{san}})$
  return 1, if $a = b$
```

Fig. 6. Strengthened Unlinkability

Definition 3 (Transparency). *A sanitizable signature scheme* SanSig *is transparent, if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Transparency}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ *given in Fig. 3 returns* 1*, is negligibly close to* $\frac{1}{2}$ *(as a function of* λ*). Here, the oracle either directly signs the expected output message (*$b = 1$*) or signs the input and then sanitizes it to the expected output (*$b = 0$*).*

Please note, in accordance with *Brzuska* et al. [8], we require that $\text{MOD}_i \subseteq \text{ADM}_i$ is true. We check this for all queries to the above oracles as otherwise a trivial attack vector exists. In particular, if $\text{MOD}_i \nsubseteq \text{ADM}_i$ yields that Sanit outputs $\perp$ then this is easily distinguishable from a "fresh" signature.

Definition 4 (Strengthened Unlinkability following *Brzuska* et al.). *A sanitizable signature scheme SanSig is* unlinkable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{SUnlinkability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 6 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of λ). Here, the adversary has to guess which of the two inputted message-signature pairs was chosen to be sanitized.*

Compare our new definition of unlinkability in Fig. 6 with the original definition of [10] depicted in Fig. 4. We altered the $\mathsf{LoRSanit}$ oracle so that it does not specify the signer by having a fixed signer key pk_{sig}. In turn, now, the adversary chooses pk_{sig}. Intuitively, this captures that unlinkability holds as long as the sanitizer is honest, even if the signer happens to be dishonest. One might argue that a malicious signer can always break any privacy or unlinkability property, as it knows which messages it signed and thus, it can always recover the originally signed message. However, there exist intermediary stages, for example, if the signer is buggy, uses weak randomness, or loses its secret key. In these cases, our definition turns out to be robust, i.e., even if the signer loses its secret key, unlinkability and therefore privacy are preserved. In a sense, we cover the equivalent of forward secrecy for the case of key exchange: there, previous sessions remain secure when long-term secrets are lost [14].

Actually, we even achieve a stronger notion of secrecy than key exchange can attain. Unlinkability is even preserved when the signer loses his secret key *before* signing the message and even when using some form of "bad" secret key specified by the adversary. In Def. 4, there are no signing and proof oracles—the reason is that the adversary can now simulate those by itself. We now prove that our new notion of unlinkability is strictly stronger than the original one by *Brzuska* et al. [10].

Theorem 1. *Any strongly unlinkable SanSig is also unlinkable. The converse is not true.*

Proof. Let SanSig be a strongly unlinkable sanitizable signature scheme. We prove via reduction that SanSig is also unlinkable. Let $\mathcal{A}$ be an adversary against the unlinkability of SanSig. Using $\mathcal{A}$, we can construct an adversary $\mathcal{B}$ against the strong unlinkability of SanSig as follows. In the beginning of the game, $\mathcal{B}$ relays the sanitizer's public key to $\mathcal{A}$ and runs the key generation algorithm of the signer and returns the signer's public key pk_{sig} to $\mathcal{A}$. The signer's secret key is used to answer all queries that $\mathcal{A}$ makes to Sign and Proof. The remaining queries are queries to the Sanit oracle, which $\mathcal{B}$ simply relays, and queries to the $\mathsf{LoRSanit}$ oracle which $\mathcal{B}$ prepends with the signer's public key pk_{sig} and queries to its own $\mathsf{LoRSanit}$ oracle. In the end, $\mathcal{A}$ returns a bit that $\mathcal{B}$ returns as well. As the simulation is perfect, $\mathcal{B}$'s advantage against strong unlinkability is as big as $\mathcal{A}$'s advantage against unlinkability.

We now separate the two unlinkability notions by constructing a (contrived) scheme that fulfills the security requirements of the original security model by *Brzuska* et al. [10] but is insecure in our new model.

Let $\mathsf{SanSig} = (\mathsf{KGen}_{\text{sig}}, \mathsf{KGen}_{\text{san}}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ be a secure sanitizable signature scheme. To obtain a counterexample, we adjust the scheme as follows:

- $\mathsf{KGen}'_{\text{sig}}$ works as $\mathsf{KGen}_{\text{sig}}$, except that it appends a 1 to the public key returned by $\mathsf{KGen}_{\text{sig}}$
- $\mathsf{KGen}'_{\text{san}}$ works as its original counterpart
- Judge', Sign', and Verify' work as their original counterparts, but cut of the last bit of pk_{sig}
- Sanit' is the same as Sanit with one exception: if pk_{sig} ends with a 0, it proceeds as normal, while cutting of the last bit of pk_{sig}. Otherwise, it outputs the *original* signature and message, instead of sanitizing it.

Our stronger attacker, that can choose pk_{sig} by running $\mathsf{KGen}'_{\text{sig}}$, now wins by replacing the trailing 1 with a 0. However, the scheme is fully secure in the original definition where an attacker was not able to influence pk_{sig}. Here, it does not matter, if the keys are indistinguishable. All other properties are not affected and are inherited from the original SanSig. Note, this scheme is also still correct as the message-signature pair left untouched by the modified Sanit' still verifies, while also $\text{MOD}_1(m_1) = \text{MOD}_2(m_2)$ yields.

As our stronger notion of unlinkability implies the original notion of unlinkability, all known implications still hold. In particular, our strengthened unlinkability also implies privacy [10].

Definition 5 (Unlinkability by *Canard* et al.). *A sanitizable signature scheme SanSig is* unlinkable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{UnlinkabilityEx}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 6 returns* 1 *is negligibly close to $\frac{1}{2}$ (as a function of λ). The basic idea is that an adversary has to guess which message-signatures pair from the two inputs was chosen to be sanitized.*

For self-containment, we introduce the notation of the complete multi-sanitizer framework and refer to *Canard et al.* [13] for a complete discussion. We have a choose-then-guess adversary, which state is denoted as *st*. The algorithms and oracles are defined as follows:
(1) $\mathsf{Setup}(1^\lambda, n, m)$ is the instance generator, where n denotes the amount of signers, while m denotes the number of sanitizers. Hence, PK contains all public keys, while SK contains all private keys. The adversary $\mathcal{A}$ also gains access to sanitization, sign, proof and all "opening" oracles:
(2) $\mathsf{FindOri}$ returns the index of the original *signer*. If the message has been sanitized, $\mathsf{FindOri}$ returns 0, as "no signer exists". It requires the opening key $\mathsf{osk}_{\text{ORI}}$.

```
$\textit{Experiment}$ $\mathsf{Pubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{sig}}(1^\lambda)$
  $(pk_{\mathrm{san}}, sk_{\mathrm{san}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $(pk^*, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathrm{sig}}, \cdot, \cdot)}_{\mathsf{Sanit}(\ldots, sk_{\mathrm{san}})}(pk_{\mathrm{sig}}, pk_{\mathrm{san}})$
    Let $(m_i, \mathrm{ADM}_i, pk_{\mathrm{san},i})$ and $\sigma_i$ for $i = 1, 2, \ldots, q$
    be the queries to and from oracle $\mathsf{Sign}$
    return 1 if
      $\forall i : (pk^*, m^*) \neq (pk_{\mathrm{san},i}, m_i)$, and
      $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\mathrm{sig}}, pk^*) = 1$, and
      $\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk_{\mathrm{sig}}, pk^*, \perp) = \mathtt{Sig}$
    Let $(m_j, \mathrm{MOD}_j, \sigma_j, pk_{\mathrm{sig},j})$ and $(m'_j, \sigma'_j)$
    be the queries to/from oracle $\mathsf{Sanit}$
    return 1 if:
      $\forall j : (pk^*, m^*) \neq (pk_{\mathrm{sig},j}, m'_j)$, and
      $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathrm{san}}) = 1$, and
      $\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathrm{san}}, \perp) = \mathtt{San}$
  return 0
```

Fig. 7. Non-Interactive Public Accountability

```
$\textit{Experiment}$ $\mathsf{Immutability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$
  $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}}) \leftarrow \mathsf{KGen}_{\mathrm{san}}(1^\lambda)$
  $(pk^*_{\mathrm{san}}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\mathrm{sig}}, \cdots), \mathsf{Proof}(sk_{\mathrm{sig}}, \cdots)}(pk_{\mathrm{sig}})$
  let $(m_i, \mathrm{ADM}_i, pk_{\mathrm{san}\,i})$ and $\sigma_i$, $i = 1, 2, \ldots, q$
  denote the queries to $\mathsf{Sign}$
  return 1, if:
    $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\mathrm{sig}}, pk^*_{\mathrm{san}}) = 1$, and
    for all $i = 1, 2, \ldots, q$ we have:
      $pk^*_{\mathrm{san}} \neq pk_{\mathrm{san},i}$, or
      $m^*[j_i] \neq m_i[j_i]$, where $j_i \notin \mathrm{ADM}_i$
      shorter messages are padded with $\perp$
```

Fig. 8. Immutability

(3) $\mathcal{CU}$ denotes the set of corrupted participants, i.e., all signers and sanitizers which secret key is known to the adversary. The adversary can gain access to the secret keys by using an implicit Corrupt oracle.
(4) Judge works as in our original definition, while it accounts for multiple signer, and sanitzers resp., and allows "partial" openings. In particular, it gets an additional parameter ORI. It outputs the index of the original signer, if the message has *not* been sanitized. See [13] for a complete discussion.

Our main observation is that it is crucial for the left-or-right oracle to check the validity of both signatures before proceeding. Else, the definition is not satisfiable and would not be satisfied by the scheme in [13]. The problem is that the left-or-right oracle receives two message-signature pairs and will sanitize one of them. If one of the signature is empty, then the sanitizing algorithm is unable to produce a valid signature because else, the sanitizer would be able to break immutability. Thus, if the adversary gives a valid message-signature pair to the oracle and a pair of a message and an empty (and thus invalid) signature, then in one case, the oracle returns $\perp$, and in the other case, the oracle returns a valid answer. Thus, the adversary can distinguish the two cases.

Definition 6 (Non-Interactive Public Accountability). *A sanitizable signature scheme SanSig is* non-interactive public accountable*, if for an empty proof $\pi = \perp$, and for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Pubaccountability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 7 returns* 1 *is negligible (as a function of λ). The basic idea is that an adversary, i.e., the sanitizer or the signer, has to be able to make the Judge decide wrongly on an empty proof $\pi = \perp$. Note, Proof always returns $\perp$ and therefore is not an oracle here.*

Definition 7 (Immutability). *A sanitizable signature scheme SanSig is immutable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathsf{SanSig}}(\lambda)$ given in Fig. 8 returns* 1*, is negligible (as a function of λ). The basic idea is, that an adversary is not able to modify non-admissible blocks, even if it is able to choose the sanitizer key pair.*

Definition 8 (Secure SanSig). *We call a SanSig secure, if it is unlinkable, immutable and non-interactive publicly accountable.*

Note, unlinkability implies privacy, while non-interactive public accountability implies accountability [11] and therefore also unforgeability [8]. Recall the following separation by *Brzuska* et al. [10], which also applies for our strengthened unlinkability definition, as the latter implies the original definition of unlinkability in [10], as we have already proven.

Theorem 2 (Unlinkability $\not\Rightarrow$ Transparency). *There exists a scheme which is unlinkable, but not transparent.*

3 Efficient Perfectly Unlinkable SanSig

We introduce the building blocks used in the construction and then give a formal algorithmic description of our construction.

3.1 Building Blocks

This section introduces the required building blocks for our construction. We require a deterministic signature scheme, unforgeable under chosen message attacks (UNF-CMA). Let $\mathcal{DS} = (\mathsf{DSKGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ be such a signature scheme. Deterministic means, that signing identical messages leads to identical signatures, if signed with the same secret key *sk*. We want to emphasize that every unforgeable signature scheme can be transformed into a strongly unforgeable and also deterministic scheme using several transformations [6,20]. An example for a standardized deterministic signature scheme is "RSASSA-PKCS-v1_5-SIGN" [26].

3.2 Algorithmic Description

Our scheme is inspired by the constructions given in [9] and [11]. It achieves unforgeability, immutability, non-interactive public accountability, perfect privacy, perfect unlinkability and sanitizer- and signer-accountability. Therefore, it meets all legal and the essential cryptographic requirements.

Construction 1 (Secure SanSig). *Let $\mathcal{DS} = (\mathsf{DSKGen}, \mathsf{DSSign}, \mathsf{DSVerify})$ be a deterministic and unforgeable signature scheme. Define the sanitizable signature scheme $\mathsf{SanSig} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Judge})$ as follows:*

Key Generation: *Algorithm KGen_{sig} generates on input of the security parameter λ a key pair $(\mathrm{pk}_{sig}, \mathrm{sk}_{sig}) \leftarrow \mathsf{DSKGen}(1^\lambda)$ of the underlying signature scheme $\mathcal{DS}$, and algorithm KGen_{san} for input λ analogously returns a pair $(\mathrm{pk}_{san}, \mathrm{sk}_{san}) \leftarrow \mathsf{DSKGen}(1^\lambda)$.*

***Signing**: Algorithm* Sign *on input* $m \in \{0,1\}^*$, sk_{sig}, pk_{san}, ADM *and computes*

$$\sigma_{\mathrm{FIX}} \leftarrow \mathit{DSSign}(\mathrm{sk}_{sig}, (0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})),$$

$$\sigma_{\mathrm{FULL}} \leftarrow \mathit{DSSign}(\mathrm{sk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

using the underlying signing algorithm. It returns:

$$(m, \sigma) = (m, (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM}))$$

***Sanitizing**: Algorithm* Sanit *on input of message* m*, (maybe empty) modification instructions* MOD*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$*, keys* pk_{sig} *and* sk_{san}*, first checks that* MOD *is admissible according to* ADM *and that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig}*. If not, it stops and outputs* $\perp$*. Else, it generates the modified message* $m' \leftarrow \mathrm{MOD}(m)$ *and computes*

$$\sigma'_{\mathrm{FULL}} \leftarrow \mathit{DSSign}(\mathrm{sk}_{san}, (1, m', \mathrm{pk}_{san}, \mathrm{pk}_{sig}))$$

and outputs $(m', \sigma') = (m', (\sigma_{\mathrm{FIX}}, \sigma'_{\mathrm{FULL}}, \mathrm{ADM}))$.

***Verification**: Algorithm* Verify *on input of a message* $m \in \{0,1\}^*$*, a signature* $\sigma = (\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and public keys* pk_{sig}, pk_{san} *first checks that* σ_{FIX} *is a valid signature for message* $(0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san})$ *under key* pk_{sig} *by checking that* $\mathit{DSVerify}(\mathrm{pk}_{sig}, (0, m[\mathrm{FIX}], \mathrm{ADM}, \mathrm{pk}_{san}), \sigma_{\mathrm{FIX}}) = 1$*. Second, it returns* 1*, if:* $\mathit{DSVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$ *or* $\mathit{DSVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$*. This declares the entire signature as valid. Otherwise it returns* 0.

***Proof:** The* Proof *algorithm always returns* $\perp$

Judge: Judge *on input of* $m, \sigma, \mathrm{pk}_{sig}, \mathrm{pk}_{san}$ *and* $\perp$ *parses* σ *as* $(\sigma_{\mathrm{FIX}}, \sigma_{\mathrm{FULL}}, \mathrm{ADM})$ *and outputs* Sig*, if:*

$$\mathit{DSVerify}(\mathrm{pk}_{sig}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$$

It returns San*, if:*

$$\mathit{DSVerify}(\mathrm{pk}_{san}, (1, m, \mathrm{pk}_{san}, \mathrm{pk}_{sig}), \sigma_{\mathrm{FULL}}) = 1$$

If none verifies, it returns $\perp$.

Theorem 3 (Our construction is secure.). *If the underlying signature scheme* DS *is UNF-CMA and deterministic, then our construction is immutable, perfectly unlinkable, perfectly private, non-interactive publicly accountable (and therefore signer-/sanitizer accountable and also unforgeable [8,11]), i.e., secure.*

The proof is delegated to Appendix A.

4 Extensions: Multiple Sanitizers and Selective Linkability

We deploy a deterministic, strongly unforgeable signature scheme to obtain unlinkability. In the following, we show that by replacing the signature scheme for the fixed blocks with certain other types of signature schemes, one obtains interesting additional features, which have not been considered yet. In this section, we extend our scheme to cope with multiple sanitizers and allow for more fine-grained control.

Multiple Sanitizer and Speed-Up. Our construction can be modified to work in the multi-sanitizer framework [13]. In particular, the signer can add a public key $pk_{\mathrm{san},i}$ for each sanitizer i. Our simple yet effective alteration once again demonstrates the generality of the underlying basic idea and its broad applicability. Additionally, all mentioned modifications impact the performance of the scheme only lightly, as they require only a constant number of additional steps per sanitizer. To improve the speed of the multi-sanitizer verification procedure, one can append a hint on the required public key to the signature. This small improvement allows skipping the need to iterate through all public keys in σ_{FIX}. This also holds when considering only one sanitizer and one signer. For a meaningful definition of unlinkability in the multi-sanitizer case, we require that only one sanitizer acts as the sanitizer in the unlinkability experiment, while we also require that the sanitizers are fixed. The latter is in conjunction with [13]. To enhance anonymity for the sanitizers, one can also use a group signature scheme for the *sanitizers*. As the original signer still uses a normal signature scheme, the signatures are clearly distinguishable and therefore remain legally recognized, following the reasoning of [34]. In certain scenarios, the weakened unlinkability definition still suffices, if it is obvious in the practical application which entity has generated the signature.

Signer Selected Linkability by Strongly Unforgeable Signatures. A strongly unforgeable signature scheme is an unforgeable signature scheme, where, additionally, it is hard to generate new signatures for previously signed messages [2]. If the signature generation of a strongly unforgeable scheme is randomized, then signatures for a message are not unique which harms unlinkability in our construction. Thus, if we de-randomize the linkable scheme by using a $\mathcal{PRF}$ [20] to generate the randomness for the signing algorithm deterministically from the input message, then the signer is able to make that signature unlinkable. In detail, we consider the $\mathcal{PRF}$-key as part of sk_{sig}. Then, if two signatures are designated to be unlinkable, the used randomness is deterministically generated by applying the $\mathcal{PRF}$ on a digest of the fixed part of the message that is generated using a collision-resistant hash function. On the other hand, if the used randomness is not generated using the $\mathcal{PRF}$, the resulting sanitized documents can be linked. Overall, the $\mathcal{PRF}$-de-randomized scheme allows the signer to decide whether signed messages shall be linkable or unlinkable. It remains to establish formally that these properties hold.

Sanitizer Selective Linkability by Randomizable Signatures. As already proposed by *Brzuska* et al., re-randomizable signatures are also suitable to achieve unlinkability [10]. Interestingly, here, a dual observation to the $\mathcal{PRF}$-case applies. Namely, the sanitizer gets to choose whether a sanitized message shall be linkable to the original one or not. Note, this feature comes with the caveat that the signer relies on good randomness added by the sanitizer. If the sanitizing process is carried out by a weak or only partially trusted device, one might prefer to opt against re-randomizable signatures and use deterministic

Table 1. Median runtime of our scheme; ℓ is the number of blocks; All in ms

	KeyGen	Sign			Sanit of 25% of ℓ			Verify			Detect		
λ \ ℓ	100/1k/10k	100	1k	10k	100	1k	10k	100	1k	10k	100	1k	10k
2.048 Bit	1,934	22	24	26	13	14	17	1	1	4	1	2	5
4.096 Bit	16,280	149	150	149	78	79	84	4	4	8	5	5	9

ones, as proposed in our construction. On the other hand, it allows the signer to delegate the decision to enable linkability for certain messages to the sanitizer.

5 Performance Measurements

To demonstrate practical usability, we implemented our construction. All tests were performed on an *Intel* T8300 Dual Core @2.40 GHz and 4 GiB of RAM, running *Ubuntu* Version 12.04 LTS (64 Bit) and Java version 1.7.0_03. For all tests, we applied our algorithms to messages with 100, 1,000 (1k) and 10,000 (10k) blocks. For any block count, we decided to fix the amount of admissible blocks to 50%, and we sanitized always 50% of the admissible blocks, i.e., 25% of all blocks. We took the median of 100 runs. We used one possible choice for a deterministic signature scheme, namely "RSASSA-PKCS-v1_5-SIGN" [26]. We utilized a single thread to calculate the signatures. Obviously, parallelization, e.g., by using CRT, will yield significant performance improvements. The results of our measurements in Tab. 1 show that our scheme keeps a very high performance. The source code is available upon request.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. ePrint Report 2011/096 (2011)
2. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
4. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
5. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
6. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)

7. Brzuska, C., et al.: Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
8. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
9. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Sanitizable signatures: How to partially delegate control for authenticated data. In: Proc. of BIOSIG. LNI, vol. 155, pp. 117–128. GI (2009)
10. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of Sanitizable Signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
11. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-Interactive Public Accountability for Sanitizable Signatures. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 178–193. Springer, Heidelberg (2013)
12. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
13. Canard, S., Jambert, A., Lescuyer, R.: Sanitizable signatures with several signers and sanitizers. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 35–52. Springer, Heidelberg (2012)
14. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
15. Caplan, R.M.: HIPAA. Health Insurance Portability and Accountability Act of 1996. Dent Assist 72(2), 6–8 (1997)
16. Chang, E.-C., Lim, C.L., Xu, J.: Short Redactable Signatures Using Random Trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
17. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
18. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: Scope of security properties of sanitizable signatures revisited. In: ARES 2013, pp. 188–197 (2013)
19. Deiseroth, B., Fehr, V., Fischlin, M., Maasz, M., Reimers, N.F., Stein, R.: Computing on authenticated data for adjustable predicates. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 53–68. Springer, Heidelberg (2013)
20. Goldreich, O.: Two remarks concerning the goldwasser-micali-rivest signature scheme. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 104–110. Springer, Heidelberg (1987)
21. Gong, J., Qian, H., Zhou, Y.: Fully-secure and practical sanitizable signatures. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 300–317. Springer, Heidelberg (2011)
22. Haber, S., Hatano, Y., Honda, Y., Horne, W.G., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS, pp. 353–362 (2008)
23. Hanser, C., Slamanig, D.: Blank digital signatures. In: AsiaCCS, pp. 95–106. ACM (2013)
24. Izu, T., Kunihiro, N., Ohta, K., Sano, M., Takenaka, M.: Sanitizable and Deletable Signature. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 130–144. Springer, Heidelberg (2009)

25. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
26. Jonsson, J., Kaliski, B.: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational) (February 2003)
27. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
28. Krawczyk, H., Rabin, T.: Chameleon Hashing and Signatures. In: Symposium on Network and Distributed Systems Security, pp. 143–154 (2000)
29. Kundu, A., Bertino, E.: How to authenticate graphs without leaking. In: EDBT, pp. 609–620 (2010)
30. Lai, J., Ding, X., Wu, Y.: Accountable trapdoor sanitizable signatures. In: Deng, R.H., Feng, T. (eds.) ISPEC 2013. LNCS, vol. 7863, pp. 117–131. Springer, Heidelberg (2013)
31. Mambo, M., Usuda, E., Okamoto, K.: and Okamoto. Proxy signatures for delegating signing operation. In: CCS 1996, pp. 48–57. ACM, New York (1996)
32. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: ASIACCS 2006, pp. 343–354. ACM (2006)
33. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. IEICE Transactions 88-A(1), 239–246 (2005)
34. Pöhls, H.C., Höhne, F.: The role of data integrity in EU digital signature legislation — achieving statutory trust for sanitizable signature schemes. In: Meadows, C., Fernandez-Gago, C. (eds.) STM 2011. LNCS, vol. 7170, pp. 175–192. Springer, Heidelberg (2012)
35. Pöhls, H.C., Peters, S., Samelin, K., Posegga, J., de Meer, H.: Malleable signatures for resource constrained platforms. In: Cavallaro, L., Gollmann, D. (eds.) WISTP 2013. LNCS, vol. 7886, pp. 18–33. Springer, Heidelberg (2013)
36. Pöhls, H.C., Samelin, K., Posegga, J.: Sanitizable Signatures in XML Signature — Performance, Mixing Properties, and Revisiting the Property of Transparency. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 166–182. Springer, Heidelberg (2011)
37. Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: On Structural Signatures for Tree Data Structures. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 171–187. Springer, Heidelberg (2012)
38. Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: Redactable signatures for independent removal of structure and content. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 17–33. Springer, Heidelberg (2012)
39. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29(2), 38–47 (1996)
40. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
41. Yum, D.H., Lee, P.J.: Sanitizable signatures reconsidered. IEICE Transactions 94-A(2), 717–724 (2011)
42. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)

A Security of Our Construction

The security proofs of our construction follow the ideas of [9] and [11]. From [8,9,11] we yield that non-interactive public accountability, as defined in [11], already implies sanitizer accountability, signer accountability and unforgeability. Moreover, unlinkability implies privacy [9]. Thus, it suffices to prove that Construction 1 is immutable, non-interactive publicly accountable and unlinkable.

Proof (of Th. 3). To increase the readability, we prove each property on its own.

- *Unlinkability.* For two messages m^0 and m^1 with identical fixed parts $m[\textsc{fix}]$, the signatures $\sigma^0_{\textsc{fix}}$ and $\sigma^1_{\textsc{fix}}$ over this part are identical, as we use a deterministic signature scheme. Moreover, the signatures $\sigma^0_{\textsc{full}}$ and $\sigma^1_{\textsc{full}}$, depending on the modifiable message parts, are not used as input for the sanitizing process. Thus, we are perfectly unlinkable and perfectly private.
- *Immutability.* Assume towards contradiction that Construction 1 is not immutable. In particular, let $\mathcal{A}$ be an efficient adversary against immutability. We construct an adversary $\mathcal{B}$ against the underlying signature scheme. The adversary $\mathcal{B}$ embeds the keys of the signature scheme as the signer's public keys. It then answers $\mathcal{A}$'s queries to the signing oracle by running the algorithm as described in Construction 1, except for signature generation under the signer's key, where $\mathcal{B}$ queries its signing oracle instead of computing them itself. The simulation is perfect. When $\mathcal{A}$ returns $(m^*, pk^*_{\text{san}}, \sigma^*)$, then $\mathcal{B}$ returns $((0, m[\textsc{fix}], \textsc{adm}, pk_{\text{san}}), \sigma_{\textsc{fix}})$ as a forgery. We now prove that $\mathcal{B}$ is successful in attacking the underlying signature scheme, if $\mathcal{A}$ is.

  Following our definition, $\mathcal{A}$ wins, if it can output a tuple $(m^*, \sigma^*, pk^*_{\text{san}})$ such that $\mathsf{Verify}(m^*, \sigma^*, pk_{\text{sig}}, pk^*_{\text{san}}) = 1$ and $pk^*_{\text{san}} \neq pk_{\text{san},i}$ for all i queries to the signing oracle or $\exists i, j, j_i \notin \textsc{adm} : m^*[j_i] \neq m_i[j_i]$.

  (i) If $pk^*_{\text{san}} \neq pk_{\text{san}i}$, then $(0, *, *, pk^*_{\text{san}})$ is fresh.
  (ii) If $\exists i, j, j_i \notin \textsc{adm} : m^*[j_i] \neq m_i[j_i]$, then $(0, m[\textsc{fix}]^*, \textsc{adm}, pk^*_{\text{san}})$ is fresh.

  These cases are equal to the attack cases for forgeries of the underlying signature scheme. Thus, $\mathcal{B}$'s success probability is equal to $\mathcal{A}$'s success probability.
- *Non-Interactive Public Accountability.* Let $\mathcal{A}$ be an efficient adversary against non-interactive public accountability. We construct another efficient adversary $\mathcal{B}$ against the unforgeability of the underlying signature scheme $\mathcal{DS}$ as follows. $\mathcal{B}$ gets as input a public key pk and flips a coin b. If $b = 0$, it sets $pk_{\text{sig}} := pk$ and runs DSKGen to generate $(pk_{\text{san}}, sk_{\text{san}})$. If $b = 1$, it sets $pk_{\text{san}} := pk$ and runs DSKGen to generate $(pk_{\text{sig}}, sk_{\text{sig}})$. To simulate the oracles for $\mathcal{A}$, the algorithm $\mathcal{B}$ runs the algorithms Sign and Sanit according to the specification with the exception that whenever a signature is generated under the secret key sk corresponding to pk, $\mathcal{B}$ does not generate the signature itself. Instead, $\mathcal{B}$ queries its signing oracle and passes the result to $\mathcal{A}$. Eventually, the adversary $\mathcal{A}$ outputs a triple (pk^*, m^*, σ^*). We distinguish between two cases, a malicious sanitizer attack and a malicious signer attack.

With probability $\frac{1}{2}$ the simulation was done for the correct case, as in both cases, the output distributions of $\mathcal{B}$'s simulation are identical.

Malicious Sanitizer
$\mathcal{B}$ returns $((0, m[\textsc{fix}]^*, \textsc{adm}, pk^*_{\text{san}}), \sigma_{\textsc{fix}})$. As $m[\textsc{fix}]^*$ is fresh, the signing oracle has never signed a message of the form $(0, m[\textsc{fix}], \textsc{adm}, *)$.

Malicious Signer
$\mathcal{B}$ returns $((1, m^*, pk_{\text{san}}, pk^*_{\text{sig}}), \sigma_{\textsc{full}})$. As m^* is fresh, the signing oracle has never signed a message of the form $(1, m^*, *, *)$.

Analysis
Thus, the overall success probability of $\mathcal{B}$ is exactly $\frac{1}{2}$ the success probability of $\mathcal{A}$. □

Revocation and Non-repudiation: When the First Destroys the Latter

Johannes Braun[1], Franziskus Kiefer[2], and Andreas Hülsing[1]

[1] TU Darmstadt, Germany
{jbraun,huelsing}@cdc.informatik.tu-darmstadt.de
[2] University of Surrey, United Kingdom
f.kiefer@surrey.ac.uk

Abstract. Electronic signatures replace handwritten signatures in electronic processes. In this context, non-repudiation is one of the most desired properties – yet in practice it cannot be provided by the signature schemes themselves. Therefore, additional mechanisms in the underlying public key infrastructure are required. In this work, we present a formal treatment of that issue. We extend the formal model for public key infrastructures by Maurer introducing transitions to make it dynamic. We use the extended model to evaluate the relationship between non-repudiation and revocation and prove that backdated revocation always destroys the non-repudiation property. We prove that forward secure signatures can be used to maintain non-repudiation, rendering the costly use of time-stamping – as required by all existing solutions – superfluous. We also show how to realize this in practice, introducing a new index reporting protocol. Moreover, we show how this protocol can be used to support detection of malicious key usage, thereby improving the overall security of electronic signing. Besides, the index reporting protocol allows for a convenient realization of pay per use models for certificate pricing.

1 Introduction

Over the past few years, the importance of eBusiness and eGovernment has been steadily growing. More and more processes are handled online without physical interaction. To guarantee for authenticity and non-repudiation in such processes, electronic signatures are used. Moreover, many countries allow to replace handwritten signatures by electronic signatures and consider these as legally binding [17]. This allows to transfer many processes to the digital world that formerly required a media disruption, e.g. in many countries applying for a bank account. However, there is a fundamental difference between handwritten and electronic signatures. While handwritten signatures are naturally bound to a single person, the binding between electronic signatures and a person is artificial and thus fragile. The private key, required to generate signatures, can be applied by anyone who knows it or has access to it, without any possibility to distinguish which signature has been generated by whom. Thus, electronic signatures can only provide authenticity and non-repudiation as long as the private signature key is

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 31–46, 2014.

only applicable by a single person. While for authentication exclusive applicability during signature generation is only checked once, for use-cases that require non-repudiation it must be provable as long as the signature is of any interest. In many cases, non-repudiation must be preserved for ten years and more by law. So far, this problem was never approached using a formal analysis. We make up for this omission. Namely, we present a formal treatment of the problem of preserving non-repudiation in practice. Additionally, we show how this problem can be solved more efficiently than today.

The binding between a specific key and a person is realized by a (hierarchical) Public Key Infrastructure (PKI). In a PKI the binding between the signer's identity (e.g. a name) and his public key is established using certificates, issued (i.e. signed) by a Certification Authority (CA). This CA is also responsible for a revocation of the certificate in case of a key compromise. This is necessary, as there does not exist any usable and perfect protection of the secret key. However, it is necessary to protect the secret key in a way that a key compromise can be detected and that the key is protected at least until it is revoked. For this reason secure storage devices like smart cards should be used. In case of legally binding signatures a secure storage device is even required by law in most countries. But even secure storage devices can not guarantee perfect security. The device can be stolen or get lost. While these devices can be assumed to protect the secret key for short time span they should not be assumed to protect the key for a long time period, i.e. years, if an adversary has direct access. The progress in cryptanalysis, especially in the field of side-channel attacks, periodically proves such assumptions wrong. Anyhow, as revocation exists, the secure storage device only has to protect the secret key until the key compromise is detected and the key is revoked.

To summarize, the binding between key and a person is only temporary, terminated either by expiry of the certificate or revocation. And this is where the non-repudiation property, which is guaranteed by the electronic signature in theory fails in reality if there are no additional measures. As compromise is possible, a key owner can simply claim that his key was compromised, ascribing the generation of signatures to an adversary and thereby repudiating valid signatures.

To prevent such a repudiation attack, a chronological order of events is required and must be considered during signature validation. A signature should then be verified as valid, if it was generated before a key compromise. In practice, the actual result of a signature validation does not only depend on the verification result of the candidate signature but also on the validation model. Validation models for hierarchical PKIs define, which certificates in the certificate chain of the candidate signature have to be valid at which time for a successful validation. The current Internet standard for certificate validation, namely the shell model [9], cannot be used for legally binding signatures, as it does not take the order of events into account [3,7] and hence a signature becomes invalid if any certificate in the chain is expired or revoked. The chain model (cf. Section 2.1) in contrast is applicable as it takes into account the chronological order of events and allows to verify a signature as valid, if all signatures in the certificate chain and on the document were generated before the corresponding certificate was revoked or expired. The crux of implementing the Chain model is to establish this provable chronological order of events.

Common signature schemes do not provide inherent properties to determine and prove the chronological order of generated signatures. Today's solution is to apply an indirection and use time-stamps generated by trusted third parties – an inefficient approach with plenty of disadvantages (see Appendix A). We propose a new solution using forward secure signature schemes (FSS) which have chronological ordering as an inherent property. But some challenges remain, namely how to establish a before and after relation between signature generation and revocation in the face of dishonest end-entities aiming at repudiating their own signatures. A similar approach was presented in [18], yet, problems like compromise detection and key update scheduling remain open. Furthermore, non-repudiation is established at the end of a time period and not directly when a verifier obtains a signature.

Contribution. Based on Maurer's formal model for PKI [15] and its extensions [14,5], we establish a generalized and dynamic PKI model, in which a before and after relationship between two events can be described. This allows us to describe non-repudiation, which was not possible in previous models. We show, that non-repudiation strictly requires the prevention of backdated revocation, proving equivalence of the two. We evaluate the problems with backdated revocation and discuss different methods to prevent it. We evaluate our model on the approach of trusted time-stamps in Appendix A.

We present a new solution to establish a provable chronological order of events based on FSS, called Sign & Report, and prove that it can be used to achieve non-repudiation. Besides that we show how the proposed solution allows for a complementary security mechanism to detect key compromises. We also present a practical realization of the Sign & Report approach. Its core is an index reporting protocol, which allows monitoring of key usage by a TTP. Compared to the current approach of trusted time-stamps, with our approach we save one online step, do not need an independent infrastructure and save the signature generation and validation of the time-stamps. Furthermore, this allows for pay per use signatures besides the targeted core functionalities.

2 Security Model

In this section we propose a new extension to the formal security model introduced by Maurer in [15]. This extension allows to model revocation and formally define the notion of non-repudiation by using transitions between states. In this work we are only concerned with malicious end-entities. For the sake of simplicity, we thus do not consider attacks against CAs. One might for example use [7] to handle these. Hence we assume CAs to be trustworthy and non compromised.

2.1 Formal PKI Model

In [15] Maurer introduced a formal security model for public key infrastructures later extended by Marchesini et al. [14] and Bicakci et al. [5]. We build our model upon [14] as they introduce a smooth notion of how to handle time. We generalize their model in the sense that we do not depend on real time, but allow any indexing that admits a chronological ordering. This still includes the usage of real time information for indexing. While all former models are static, meaning they model one snapshot of a PKI, we introduce transitions between snapshots of the PKI, making the model dynamic. We further extend the model of [14], adding an explicit definition of revocation handling and end-entity signatures. This allows us to discuss non-repudiation using our model.

For simplicity, we only model relations starting from Sub-CAs that sign end-entity certificates. Thus, we assume trust in these Sub-CAs without considering how this trust is established. As we are only concerned with malicious end-entities, this fulfills our needs. It is straight forward to extend our model and proofs to the case of a hierarchical PKI. We also drop those parts of former models used to model a web of trust.

We model a PKI as View of a potential user at a specific time t. A user's View is a set of statements. We define six different statements. Trust expresses the trust in a (Sub-)CA, obtained according to the higher hierarchy or by explicitly trusting this CA. Cert says that the user has seen an end-entity certificate of the respective person. If a user has seen a certificate once, it remains in her view. The same holds for $\mathsf{Signature}$ and Revoc, which model that a user has seen a document signature or revocation information, respectively. Furthermore, there are two different Valid statements, which model that a user is convinced of the validity of an end-entity's certificate $C_{\alpha,\beta,\gamma,\varepsilon}$ or document signature $S_{\zeta,\eta,\delta}$. These two Valid statements can be inferred from other statements, using inference rules defined later. As we allow transitions between Views, every View is indexed with a time $t \in \mathbb{N}$. Note that indices used inside statements might be independent from the indices of the views. We write View^t for the View at time t and View if no specific t is needed.

Definition 1 (Statements). *Let* CA *denote a (Sub-)CA, $\mathcal{A}$ an end-entity's identity, D a document and $\mathcal{I}$ a (time) interval. A* $\mathsf{View}^t = \{\mathsf{stmt}_1, \ldots, \mathsf{stmt}_n\}$ *at point in time t consists of $n \in \mathbb{N}$ statements* stmt_i. *There exist the following six statements:*

$\mathsf{Trust}(\mathrm{CA}, \mathcal{I})$ *denotes the belief that, during the interval $\mathcal{I}$,* CA *is trustworthy for issuing certificates, i.e. models the axiomatic trust in (Sub-) CAs.*

$\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i, \mathcal{I})$ *denotes the fact that* CA *has issued a certificate for $\mathcal{A}$ at index i, which, during $\mathcal{I}$, binds $\mathcal{A}$'s public key to the certificate.*

$\mathsf{Signature}(\mathcal{A}, D, i)$ *denotes the fact that $\mathcal{A}$ has signed a document D at index i.*

$\mathsf{Revoc}(\mathrm{CA}, C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the fact that* CA *has revoked the certificate $C_{\alpha,\beta,\gamma,\varepsilon}$, represented by statement* $\mathsf{Cert}(\alpha, \beta, \gamma, \varepsilon)$, *at index i.*

$\mathsf{Valid}(C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the belief that certificate $C_{\alpha,\beta,\gamma,\varepsilon}$ is valid at evaluation index i.*

$\mathsf{Valid}(S_{\zeta,\eta,\delta})$ *denotes the belief that signature $S_{\zeta,\eta,\delta}$, represented by statement* $\mathsf{Signature}(\zeta, \eta, \delta)$, *is valid.*

A statement is `valid` if and only if it is in the View or can be derived from it using one of the inference rules defined below.

Signature Validation. We will now define the inference rules we use to validate signatures, i.e. derive `valid` for a $\mathsf{Signature}$. The rules depend on the used certification path validation model. As mentioned in the introduction, we use the chain model shown in Fig. 1 instead of the current Internet standard (shell model). This allows us to exploit the chronological ordering of signatures as e.g. provided by FSS. In the chain model all signatures in the chain are validated at the time of their generation, meaning revocation and certificate validity is checked for that time. To describe the shell model to analyze other scenarios, different inference rules have to be defined. For a detailed discussion of validity models see e.g. [3,7].

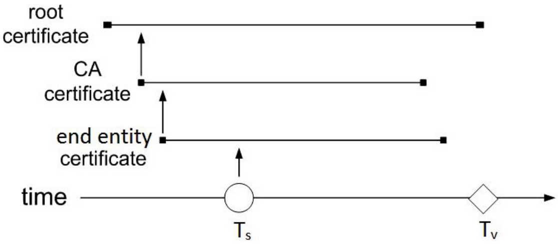

Fig. 1. Chain Model. Signature Generation at T_s, Signature Verification at T_v. Vertical arrows show the point in time used for validation of the superordinate certificate.

Definition 2 (Inference Rules). *Statements can be derived from an existing* View^t *according to the following rules:*

Certificate Validity. $\forall$ CA, $\mathcal{A}, i_r \leq i_v, i_c \in \mathcal{I}_1, i_v \in \mathcal{I}_2$: $\mathsf{Trust}(\mathrm{CA}, \mathcal{I}_1)$, $\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i_c, \mathcal{I}_2), (\neg\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)) \vdash \mathsf{Valid}(C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_v)$

Signature Validity. $\forall$ CA, $\mathcal{A}, D, i_s \in \mathcal{I}_2$: $\mathsf{Valid}(C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_s), \mathsf{Signature}(\mathcal{A}, D, i_s) \vdash \mathsf{Valid}(S_{\mathcal{A},D,i_s})$

Note, to extend the model to certificate chains of arbitrary length, one simply considers certificates as signed documents while processing the chain, and derives the validity accordingly.

So far our model is static. To allow the definition of non-repudiation we introduce *transitions* between Views. The transitions model that new information enters a users View in form of certificates, signatures or revocation information. Besides that, a user might trust a new (Sub-)CA.

Definition 3 (Time & Transitions). *Let* View^t *be the* View *at time t and* $\mathsf{View}^t \xrightarrow{\texttt{trans}} \mathsf{View}^{t+1}$ *denote the transition from* View^t *to* View^{t+1}. *Let CA denote a (Sub-)CA, $\mathcal{A}$ an end-entity's identity, D a document and $\mathcal{I}$ an interval. We allow the following four transitions between* Views*:*

- $\mathsf{View}^t \xrightarrow{\mathsf{sign}(\mathcal{A},D,i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Signature}(\mathcal{A}, D, i)$ *to* View.
- $\mathsf{View}^t \xrightarrow{\mathsf{issue}(\mathrm{CA},\mathcal{A},i,\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i, \mathcal{I})$ *to* View.

- $\mathsf{View}^t \xrightarrow{\mathsf{trust}(\mathrm{CA},\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Trust}(\mathrm{CA},\mathcal{I})$ *to* View.
- $\mathsf{View}^t \xrightarrow{\mathsf{revoke}(\mathrm{CA},C_{\alpha,\beta,\gamma,\varepsilon},i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Revoc}(\mathrm{CA},C_{\alpha,\beta,\gamma,\varepsilon},i)$ *to* View.

Please note that derived statements are temporary. After a transition between two views, the inference rules are used again, to obtain the full set of statements. With $\overline{\mathsf{View}}$ we denote the set of all statements that can be inferred from View. So, if $\mathsf{stmt} \in \overline{\mathsf{View}^t}$ it does not have to be the case that $\mathsf{stmt} \in \overline{\mathsf{View}^{t'}}$ for $t \neq t'$. For example, if a certificate gets revoked, Valid might be inferable beforehand but not after Revoc has been added to the View.

2.2 Non-repudiation

In this work we assume adversaries that try to break the non-repudiation property and to which we refer as *repudiation adversaries*. The goal of such an adversary is to retroactively invalidate a signature, validly generated by herself. The adversary has access to the corresponding key pair, including the secret key and the certificate. The adversary might also have different other key pairs and certificates, possibly issued by other CAs. We give the classic non-repudiation definition [12] in our model. This allows a more precise analysis of repudiation adversaries.

Definition 4 (Non-Repudiation). *A PKI offers non-repudiation if the following implication is always true, even in presence of a malicious end-entity that might sign arbitrary messages, request new certificates and ask any CA to revoke any of her certificates at anytime.*

$$\forall\, i,\ t \leq t' \ :\ \mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^t} \Rightarrow \mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^{t'}}.$$

We briefly discuss the implications of this definition. The left part of the implication – $\mathsf{Valid}(S_{\mathcal{A},D,i}) \in \overline{\mathsf{View}^t}$ implies that

$$\{\mathsf{Signature}(\mathcal{A},D,i), \mathsf{Trust}(\mathrm{CA},\mathcal{I}_1), \mathsf{Cert}(\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2)\} \subseteq \overline{\mathsf{View}^t}$$

with $i_c \in \mathcal{I}_1,\ i \in \mathcal{I}_2$ according to the previously given inference rules and definitions. Furthermore, $\mathsf{Revoc}(\mathrm{CA},C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2},i_r) \notin \overline{\mathsf{View}^t}$ for all $\mathcal{I}_2 \ni i_r \leq i$. In other words, three things must be in View^t: (i) trust in the certification authority CA that issued the end-entity certificate for the document signing entity $\mathcal{A}$, (ii) the certificate of $\mathcal{A}$ that has been issued while CA has been trusted, (iii) a signature on the verified document D that has been issued by the end-entity $\mathcal{A}$ while his certificate has been valid, i.e. was not revoked or expired. The right part of the implication only differs in the time of inference of the Valid statement. Thus, everything above must hold for all future points in time t'.

Accordingly, the goal of the adversary is to produce a valid document signature $\mathsf{Signature}(\mathcal{A},D,i)$ such that there exists a point in time t' where the signature is verified as invalid, after it has been verified as valid. Therefore, we define backdated revocation and show, that its prevention implies non-repudiation and vice versa in the chain model.

Definition 5 (Backdated Revocation). *Let* View^t *be the* View *at time* t *and* View^{t+1} *denote the view after a transition. According to the revocation transition, backdated revocation is defined as:*
$\mathsf{View}^t \xrightarrow{\mathsf{revoke}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I},i_r})} \mathsf{View}^{t+1}$, *if* $\exists\ \mathsf{View}^{t^*} \ni \mathsf{Valid}(S_{\mathcal{A},D,i_s})$, *with* $t^* \leq t \wedge i_s \geq i_r$.

Theorem 1 (Non-Repudiation ⇔ No Backdated Revocation). *A PKI offers non-repudiation according to Definition 4 if and only if it does not allow backdated revocation according to Definition 5.*

Proof. ⇐: If there was a successful repudiation attack, then there must exist two Views $\overline{\mathsf{View}^t} \supseteq \{\mathsf{Valid}(S_{\mathcal{A},D,i_s}), \mathsf{Trust}(\mathrm{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i_c, \mathcal{I}_2)\}$ and $\overline{\mathsf{View}^{t'}} \supseteq \{\mathsf{Trust}(\mathrm{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathrm{CA}, \mathcal{A}, i_c, \mathcal{I}_2), \mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)\}$, with $t \leq t', i_r \leq i_s$. As $\mathsf{Valid}(S_{\mathcal{A},D,i_s})$ is contained in $\overline{\mathsf{View}^t}$, it can not contain $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$. Hence, $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$ must have been added later, which exactly corresponds to Definition 5.
⇒: If the PKI allows backdated revocation, the adversary is allowed to ask CA to add $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}_2}, i_r)$ with $i_r \leq i_s$ to the $\mathsf{View}^{t'}$. □

3 Enabling Non-repudation

Theorem 1 directly shows the impossibility to achieve non-repudiation if backdated revocation is allowed. Hence, a PKI that offers non-repudiation *must not* allow backdated revocation. However, when considering the facets of backdated revocation there are different security goals that contradict each other. This conflict needs to be resolved, and is discussed in the following.

Contradicting Security Goals. We have learned from our formal model, that there is no way to guarantee non-repudiation in case backdated revocation is allowed. On the other hand, backdated revocation is required in certain scenarios, namely whenever it is possible that the signature key might get compromised and maliciously used without being noticed immediately by the key owner. For example consider a classical setup for digital signatures where the private key is stored on the user's machine (e.g. PC). Here, the detection of a key compromise may take some time in which the adversary who stole the key may already have generated signatures. The phase between key compromise and the observation of the compromise (followed by the revocation) is called *gray phase*. The gray phase in this case however can be very long, such that it is clearly impossible to prohibit backdated revocation. This is because backdated revocation is required to invalidate the signatures generated by the adversary before the compromise detection.

Thus, in order to be able to prohibit and subsequently prevent backdated revocation, scenarios with a (possibly large) gray phase must be excluded or the gray phase must be eliminated by technical means. Therefore, in scenarios that require non-repudiation, the secret key has to be protected in a way that prevents unnoticed compromises. A common solution that allows for a minimal gray phase is to store keys on smart cards, trusted platform modules (TPM) etc. Private keys are not extractable from these devices and can only be used when the according secret, e.g. PIN, is known. This allows for the assumption of "immediate" key compromise detection and subsequently the prohibition of backdated revocation in the sense that an adversary is not able to immediately crack the additional secret (e.g. PIN) and thus use the stolen key before the key compromise, i.e. disappearance of the key storage device, is detected. Note, that in Section 5 we show how our solution supports immediate compromise detection for end-entities. This further justifies the assumption of a marginal gray phase. In general, backdated revocation is not necessary when the gray phase is marginal.

How to Prevent Backdated Revocation. With the aforementioned it is sufficient and legitimate to prohibit backdated revocation. Nevertheless, the question remains how to enforce the prohibition in practice, i.e. how to implement our model. We stick to the PKI setting and hence we assume, that the CA as TTP is the only entity that can revoke certificates. Therefor it must be possible for the CA to ensure that the revocation only invalidates signatures that were not assumed to be valid before. This can either be done by explicitly defining the views $View^t$ and binding each signature, as well as the revocation, to a certain view, using a globally visible and unique index t for views and signatures. Or second, to define $View^t$ implicitly by maintaining a local chronological order among the signatures made with one key using a local immutable index. By binding the revocation to the current index, it is correctly linked into the chronological order of the signatures. Note, that this requires the responsible CA to know the correct current local index.

The first approach is followed by the application of time-stamps, using real time as the global index. The time-stamps are generated by TTPs called Time-Stamping Authorities (TSA). This approach is widely accepted to be correct but also comes with a lot of inefficiencies. To evaluate our model, we proof that the time-stamping approach allows to implement our model s.th. the resulting PKI provides non-repudiation. The proof can be found in Appendix A together with a discussion of its inherent inefficiencies.

For our new solution we use the second approach of local indexing, which can be realized using forward secure signature schemes. This is described in the following section.

4 Sign and Report

In this section, we propose our solution applying local indexing to enforce non-repudiation using forward secure signature schemes (FSS). It does not have the disadvantages and inefficiencies of the TSA approach (cf. Appendix A). Furthermore, we show how compromise detection for end-entities can easily be incorporated to prevent gray periods.

As some reader might not be familiar with the concept of FSS, we first informally recall the related definitions and discuss some properties of such schemes more detailed. For a formal treatment we refer the reader to [4]. We will not discuss the definitions regarding traditional signature schemes like existential unforgeability under chosen message attacks (EUF-CMA). Here we refer the reader to [11]. Forward security can only be achieved using key evolving signature schemes, which will be explained first. Afterwards, we show how to implement the PKI model using FSS, such that backdated revocation can efficiently be prevented.

Key Evolving Signature Schemes. Once generated, a key pair remains unchanged for the whole lifetime, in a traditional signature scheme (SIG). In contrast, the secret key $\mathtt{sk}$ changes over time, while the public key $\mathtt{pk}$ remains the same, in a key evolving scheme (KES). More specific, the lifetime of a key pair is split into several time-periods, say T. The number of time-periods T becomes a public parameter of a KES and is taken as an additional input by the key generation algorithm. The key generation algorithm outputs $(\mathtt{sk}_0, \mathtt{pk})$, where $\mathtt{sk}_0$ is the first secret key. In contrast to SIG, a KES has an additional key update algorithm, which updates the secret key at the end of each time period. The sign algorithm takes as an additional input an index of a time-period. This index also becomes part of the signature and is therefore also available for the verify algorithm. Finally, if a user generates a valid key pair and the key update algorithm is called at the end of each time-period, then a signature generated with the current secret key and the index of the current time period can be verified by any user with the corresponding public key.

Forward Secure Signature Schemes. A forward secure signature scheme (FSS) is a KES that provides the forward security property. The main idea behind forward security is that even after a key compromise all signatures created before should remain valid. The forward security property guarantees, that an adversary that is allowed to launch a chosen message attack for each time-period and learns the secret key of an adaptively chosen time-period i is unable to produce a valid forgery for time-period $j < i$. Note that forward security implies the standard notion of EUF-CMA extended to the case of KES. To make use of the forward security property in practice, a certificate is revoked from the index of the current time period onwards instead of revoking the complete certificate [7] in the case of key compromise.

Defining the Time Periods for FSS. One issue with FSS in practice is how the key update algorithm is triggered. It can either be called manually by the user, scheduled to run at the end of the time period, or be part of the signature algorithm, depending on the way the time periods are defined. Time periods can be defined in terms of time, e.g. one time period corresponds to one day, or the number of created signatures, i.e. a time period ends after the key was used to create a certain number of signatures. In the first case, the key update algorithm must be triggered periodically. This can only be automated on systems that have an internal clock and that are running each time an update is necessary.

On smart cards, which are the common place to store end-entity signature keys, a manual update is required. This seems impractical. As an on-time key update is required to preserve forward security, real time based time periods are problematic in practice. FSS schemes based on the number of generated signatures do not have these problems. Key update can be performed automated, based on a counter contained in the key holding device. Yet, in this case, the key indices are not linked to real time, which complicates correct revocation in practice as the index at the time of compromise must be known. However, our solution shows that this is achievable compared to the key update problem. Thus, we consider FSS where the periods are based on the number of signatures, namely an FSS, that evolves the key after each signature generation, as e.g. XMSS [8].

Sign and Report. We now show how to use a FSS, where a key update is performed after each signature, to implement a PKI according to our model, that efficiently prevents backdated revocation. If we used an FSS with real time based key update, the implementation would be straight forward, similar to the time-stamping approach. As we use an FSS where the key update occurs after each signature because of the discussed drawbacks of real time based key updates, things are a bit different. The validity periods of certificates, as well as the time interval for the trust relation are now described on the basis of indices. In general, these intervals will be $[0, T-1]$ for a key pair with T time periods. The indices used in Cert, Signature, Revoc and Valid statements correspond to key indices. For the first two types of statements, it is the current index of the key pair used to generate the signature. For the Revoc statement, it is the index starting from which on a key pair is revoked and for the last one it is the index at which revocation is checked. The indices of the views use real time. Please note that it is possible to additionally use real time validity periods, if this is required, e.g. for business purposes.

To prevent backdated revocation in such a PKI, a revocation must include the current index of an end-entity's key pair. Therefore the responsible CA must know this current index and be able to verify the correctness of this index. To achieve this, we define a Sign & Report approach for FSS.

Definition 6 (Sign & Report PKI). *A Sign & Report PKI implements the model defined in Section 2 replacing the abstract indices and intervals as described above. Let* $\mathtt{R}$ *denote a trusted third party in the PKI, e.g. a CA, which is responsible (and exclusively able) to issue the revocation of an end-entity* $\mathcal{A}$*'s certificate* $C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}}$*, when requested by* $\mathcal{A}$*. Whenever* $\mathcal{A}$ *generates a signature, the used key index* i^* *is reported to* $\mathtt{R}$ *that stores* i^**. On input of revocation request by* $\mathcal{A}$*,* $\mathtt{R}$ *publishes* $\mathsf{Revoc}(\mathrm{CA}, C_{\mathrm{CA},\mathcal{A},i_c,\mathcal{I}}, i^* + 1)$*.*

We next show that a Sign & Report PKI provides non-repudiation, assumed that the index reporting is secure.

Theorem 2 (Sign & Report PKIs provide non-repudiation). *A Sign & Report PKI as defined above provides non-repudiation according to Definition 4.*

Proof. If the index reporting is implemented in a secure way, i.e. it is not possible for an end-entity to manipulate the reporting, backdated revocation is efficiently prevented. This is the case, because the index used for revocation is greater than any index used by this end-entity before. The non-repudiation property follows from Theorem 1. □

Key Compromise Detection. The Sign & Report PKI makes it easily possible to monitor key usage and support end-entities in the detection of malicious key usage and subsequent revocation. Therewith, the justification for immediate revocation can be strengthened. The detection is made possible as, different from the TSA approach, for each end-entity a specific TTP is responsible.

Definition 7 (Sign & Report PKI with Compromise Detection). *A Sign & Report PKI with Compromise Detection is a Sign & Report PKI as defined in Definition 6 with the additional measure, that* R *requests a reconfirmation from* $\mathcal{A}$ *using an independent channel, whenever a new key index* i^* *is reported.* R *only accepts the new index if the reconfirmation succeeds. Otherwise,* R *publishes* $\mathsf{Revoc}(\text{CA}, C_{\text{CA},\mathcal{A},i_c,\mathcal{I}}, i^*)$

5 Implementing Sign and Report

In this section we present a practical implementation of the Sign & Report approach from above. More specifically we show how to securely implement the index reporting, as everything else is straight forward. Note that we apply an FSS, namely XMSS, that evolves the key after each signature generation. Thus, each signature is directly linked to a unique index. We also present extensions that allow protection from undetected key access and a pay per use pricing model for certificates. As discussed before, we assume that the private key is stored on a smart card and therefore can assume immediate revocation (cf. Section 3). With this preliminary, we define an online protocol to prevent backdated revocation.

Index Reporting Protocol. The basic idea of Sign & Report is that the current index is reported to the CA (or some TTP that maintains the revocation) immediately after signature generation. This requires an online step, yet seams reasonable, as most of today's computers are nearly always online. Thereby, the CA is enabled to keep track of the signing index and prevent the key owner from backdated revocation and repudiation of signatures. Thereby, it does not matter who reports the index, the verifier or the signer itself and this might be chosen depending on the specific application. In the first case, reporting can be performed in one combined step during online revocation status checking and would reflect the natural ambition of the verifier to obtain non-repudiation. The second case is desirable when the verifier is offline. However, then the signer needs to be able to prove the reporting. This can be realized by a *validity token* obtained from the CA and additionally serving as a proof for the absence of a revocation. Thus, additional revocation checking is made obsolete.

Figure 2 shows the protocol for the second case, but the adaptation to the first is straight forward. After signature generation (step (a)) the signature is sent to the CA (step (b)). The CA checks the signature for validity (step(c)) and generates a *validity token* for the signature index to confirm the logging.

	CA		Signer		Verifier
(a)			$\sigma = \mathsf{Sign}(\mathtt{sk}, m)$		
(b)		$\xleftarrow{\sigma}$			
(c)	If $\mathsf{Verify}(\mathtt{pk}, \sigma)$ and $\mathtt{pk}$ is not revoked: generate proof π				
(d)		$\xrightarrow{\pi}$			
(e)				$\xrightarrow{\sigma,\pi}$	
(f)					If $\mathsf{Verify}(\sigma, m)$ and
(g)					$\mathtt{Verify}(\pi)$: accept

Fig. 2. Index Reporting Protocol

Herein, the signature verification ensures, that the report and confirmation of wrong index information is prevented. The token is sent back to the signer and subsequently transmitted (together with the signature) to the verifier (steps (d)-(e)), who can now validate the signature and the token.

By the index, the validity token is bound to a specific signature. Thus, it can be used for all future verifications without further online requests. Additionally, due to the forward security, the token for a certain index i can serve as a validity token for all preceding indices. Thus, if several signatures have to be validated, the logging request can be aggregated to only one, by requesting the token for the highest index.

The validity token can be realized as public key - index pair signed by the CA. One drawback of such a signed validity token is that the signature on the token has to be validated in addition to the validation of the end-entity signature. Furthermore, the size of the token is not optimal and signature generation is computationally complex. We propose to apply the Merkle tree variant of the Novomodo system [16] to generate the validity token (cf. Appendix B for details).

Incorporation of Compromise Detection. In the index reporting protocol, the issue of compromise detection and the prevention of gray periods can be addressed by adding an additional reconfirmation procedure for signature generation. Before confirming the logging of the index, the CA can request a reconfirmation from the key owner. A possibility to do so is to apply mobile transaction numbers as known from eBanking or similar to the usage presented in [6]. This helps to detect malicious key usage, as the key owner is informed about key usage via an independent channel. By additionally sending the document itself to the CA, the signed document could be sent back and displayed on a smartphone for verification. Therewith, undetected usage of the key is significantly less probable, and even cases where e.g. the smart card is left unwatched for a certain time period and malicious key usage does not necessarily involve the observed loss of the card can be detected.

Efficiency. Besides cutting down on the maintenance of an additional and independent TSA infrastructure and the overhead of time-stamp validation, our approach saves one online request considering the whole process from signature generation to verification. While in the TSA approach the time-stamp and the revocation status need to be requested, one online request, namely the index logging request, is sufficient in our approach.

6 Conclusion

In this work we showed how to extend the existing formal models for a PKI such that it becomes possible to describe the non-repudiation property. Using our model, we proved that non-repudiation is achieved, if and only if backdated revocation is prevented (assuming CAs are trustworthy and the used cryptographic algorithms are secure). The main improvement of our model is that it is dynamic. It might also be useful in analyzing other properties of PKIs. Furthermore, we showed how to realize our model using FSS. We presented an index reporting protocol to implement this new approach. It has some clear advantages compared to today's solutions regarding computation costs and storage. Furthermore, it allows for a convenient integration of an additional reconfirmation step to detect compromises and improve the overall security. Besides that, the index reporting approach allows for another interesting application we shortly want to mention, namely the realization of a pay per use pricing model. This is possible, as the CA is enabled to monitor the frequency of key usage. Thus, the costs for a certificate can be spread over the whole key lifetime, therewith the costs to get a certificate can be lowered significantly. This might on the one hand help to decrease the initial barrier of buying a certificate for end-entities that rarely apply electronic signatures. On the other hand revenues for the CAs are generated at the time when the efforts arise for the management of certificates.

While Sign & Report improves the performance of document signatures it still requires one online step per signature, i.e. the signer has to communicate with a TTP during signature creation (This step can be shifted to the verifier, but it remains one online step per signature). We were unable to get rid of this costly step. Hence the question arises if it is possible at all to prohibit backdated revocation by using an "offline" solution, namely a solution where the end-entity does not report information about each signature to a TTP? As we assume the signer as well as the verifier to be potentially malicious, neither of them can be trusted. A trusted device in possession of the signer or verifier can also not be trusted, as it might in the long term be possible to tamper with or it might get destroyed. In both cases, i.e. using a local or a global index, the indexing – especially the order – and the linking between indices and signatures must be immutable. Furthermore, using local indexing the CA must be able to obtain the correct current value of the index used by the user in case of a revocation. Given these constraints it seems impossible to us to solve this problem without an online step. At least with existing techniques we see no solution to this challenge.

References

1. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: RFC 3161 — Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard) (2001), Updated by RFC 5816

2. ANSI X9.95-2012 — Trusted Time Stamp Management and Security (2012)
3. Baier, H., Karatsiolis, V.: Validity models of electronic signatures and their enforcement in practice. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 255–270. Springer, Heidelberg (2010)
4. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
5. Bicakci, K., Crispo, B., Tanenbaum, A.S.: How to incorporate revocation status information into the trust metrics for public-key certification. In: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC 2005, pp. 1594–1598. ACM (2005)
6. Braun, J., Horsch, M., Wiesmaier, A.: iPIN and mTAN for secure eID applications. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 259–276. Springer, Heidelberg (2012)
7. Braun, J., Hülsing, A., Wiesmaier, A., Vigil, M.A.G., Buchmann, J.: How to avoid the breakdown of public key infrastructures - forward secure signatures for certificate authorities. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 53–68. Springer, Heidelberg (2013)
8. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - A practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011)
9. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Rfc 5280 — internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280 (Proposed Standard) (2008)
10. Elwailly, F., Gentry, C., Ramzan, Z.: QuasiModo: Efficient Certificate Validation and Revocation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 375–388. Springer, Heidelberg (2004)
11. 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)
12. ISO/IEC 13888-1 — Information technology-security techniques-non-repudiation, Part 1 (1997)
13. ISO/IEC 18014 — Information technology – Security techniques – Time-stamping services (2009)
14. Marchesini, J., Smith, S.: Modeling public key infrastructures in the real world. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 118–134. Springer, Heidelberg (2005)
15. Maurer, U.M.: Modelling a Public-Key Infrastructure. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 325–350. Springer, Heidelberg (1996)
16. Micali, S.: NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In: 1st Annual PKI Research Workshop, PKI 2002, pp. 15–25 (2002)
17. The European Parliament and the Council of the European Union. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. In European Union law. The European Parliament and Council (1999)
18. Zhou, J., Bao, F., Deng, R.: Minimizing ttp's involvement in signature validation. International Journal of Information Security 5(1), 37–47 (2006)

A Application of Time-Stamping Authorities

To evaluate our model, in this section we show that the common approach of time-stamping allows to implement a PKI that provides non-repudiation. To implement a PKI according to the above model using time-stamping, a trusted third party called Time-Stamping Authority (TSA) adds a trusted time to each signature. This can be implemented in different ways [1,2,13]. The most common one is to sign the signature together with a time-stamp. The signature time of a document signature as well as the issuance time of a certificate are given by a time-stamp. Validity periods of certificates are defined by the issuing CA and the time interval for the trust statement is defined by the user (or according to the certificates of higher levels in the hierarchy). Both are defined in terms of real time. If a CA is asked to revoke a certificate, it uses the current time as revocation index. All views are also bound to real time. So all indices and intervals are directly linked to real time and an order is defined according to the calendar.

Definition 8 (Sign & Stamp PKI). *A Sign & Stamp PKI implements the above model using the real time for all indices and intervals. In a Sign & Stamp PKI, every signature is time-stamped by a trusted third party called time-stamping authority (TSA). If a CA is asked to revoke a certificate, it uses the current time as revocation index.*

In a Sign & Stamp PKI times in statements and Views have a strictly monotonic increasing order, e.g. a signature generated at time t can not be in a $\mathsf{View}^{t'}$ with $t' < t$. As revocations always include the current time, backdated revocation is impossible (recall that we assume the used signature schemes to be perfectly secure). The following theorem follows immediately from Theorem 1 and the fact that a Sign & Stamp PKI prevents backdated revocation.

Theorem 3 (Sign & Stamp PKIs provide Non-Repudation). *A Sign & Stamp PKI according to Definition 8 provides non-repudiation according to Definition 4.*

Disadvantages of the TSA Approach. We have seen that the TSA approach enables non-repudiation. Yet, it comes with many disadvantages. First, the setup and maintenance of an additional and independent TSA infrastructure and the trustworthiness of the TSAs to apply the correct date and time is required. Second, it introduces overhead during signature generation (for the online request and generation of the proof of existence) and during validation (for the proof validation). Third, storage overhead is introduced, i.e. time-stamps or MACs must be stored, in addition to the signature itself, or a huge amount of transient keys must be managed. Database based approaches require the central storage of all issued signatures. And fourth, time-stamps relying on electronic signatures themselves face the same problems concerning compromise and expiration as common electronic signatures do. That is, upon the compromise of a TSA or any superordinate CA, the issued time-stamps become invalid and the proof of existence is lost. On the other hand, database based approaches solely rely on the security of the central database.

B Validity Tokens Using Novomodo

Instead of using hash chains as in the basic Novomodo system [16] one can use Merkle hash trees. This significantly improves the verification time. To realize the validity tokens, the root of a Merkle tree with T leaves (where T is the number of periods, the respective key pair is valid for) is included into the certificate. If the certificate is valid in period i, the CA releases the leaf at position i and the siblings on the path to the root. QuasiModo trees [10] allow even smaller trees and on average shorter paths to the tree root by using interior nodes, yet for standard Merkle trees highly efficient traversal methods are available [8]. The validity tokens of the revocation tree can be generated in a pseudorandom fashion, comparable to the approach used for XMSS key generation [8]. An application of the hash function gives us the leaves from which the Merkle tree is computed. Furthermore, the tree traversal algorithm from [8] can be used to evenly split the computational effort over all periods. To prevent delays, a certain number of validity tokens can be precomputed and stored, reducing the effort to a table lookup during the online request.

New Results for the Practical Use of Range Proofs

Sébastien Canard[1], Iwen Coisel[2], Amandine Jambert[3], and Jacques Traoré[1]

[1] Orange Labs
Applied Crypto Group
42 rue des coutures
14000 CAEN, France
{sebastien.canard,jacques.traore}@orange.com

[2] European Commission - Joint Research Centre (JRC)
Institute for the Protection and the Security of the Citizen
Digital Citizen Security
21027 ISPRA (VA), Italy
iwen.coisel@jrc.ec.europa.eu

[3] CNIL - Commission Nationale de l'Informatique et des Libertés
8, rue Vivienne - CS 30223
75083 PARIS cedex 02, France
ajambert@cnil.fr

Abstract. Zero-knowledge proofs of knowledge are now used in numerous applications and permit to prove the knowledge of secrets with many (complex) properties. Among them, the proof that a secret lies in a given interval is very useful in the context of electronic voting, e-cash or anonymous credentials. In this paper, we propose new contributions to the practical use of these so-called *range proofs*, for which several types of methods exist. We first introduce a variant of the *signature-based* method which allows the prover to avoid pairing computations. We also give several improvements to the solution based on the multi-base decomposition of the secret. We finally make the first complete comparison between all existing range proofs. This permits to prove that our methods are useful in many practical cases. This also allows service designers to decide which method is the best to use in their case, depending on their practical needs and constraints on the size of the interval, the power of the verifier and the prover, etc.

Keywords: Range proof, set membership proof, binary decomposition, signature based method.

1 Introduction

In the authentication context, zero-knowledge proofs of knowledge (ZKPK) are largely used *e.g.* to prove the possession of some secrets corresponding to a discrete logarithm [35], of a representation [32] or the equality of secrets [17]. In some particular cases, the prover may need to show that her underlying secrets

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 47–64, 2014.

verify some additional properties. In this paper, we focus on the so-called *range proofs* which permit to prove, in a zero-knowledge way, that a secret lies in a given and public interval.

Range proofs are useful in many cryptographic applications. E-cash and multi-coupon systems [12,13] sometimes require to prove that the secret counter j of already spent coins is not greater than the number J of withdrawn coins. It is also very useful for anonymous credential systems [11] where a user may have to prove *e.g.* that her unrevealed age is greater than a public threshold. They are also used in the context of electronic voting [26,3], where the voter needs to prove that her secret vote belongs in the set of all possible candidates. In this case, we generally speak of *set membership proofs* (instead of range proofs).

The first proposals [9,16] for ZKPK of a secret lying in a specific interval were very efficient but they only prove the membership to a slightly larger interval than expected. In this paper, we focus on exact range proofs. There now exists many possibilities to design such exact range proof and we are today able to clearly describe the fundamental advantages and drawbacks of each family.

The first type of method, introduced by Boudot [8], uses some mathematical properties of positive integers such as its decomposition in a sum of squares. This method was later refined by Lipmaa [27] and next by Groth [24] to obtain a very efficient exact method where time and space complexities are independent of the secret size. The main drawback of these methods is the need for a group of unknown order, which implies to manipulate bigger variables. Thus these methods are more adapted for the case of big secrets. Recently, one proposal of non-interactive proof, secure in the standard model, has been proposed in [15], based on such kind of method.

The second family [4,19,36,28] uses the decomposition of the secret in a (multi)-base. Bellare and Goldwasser [4] have been the first to use the binary decomposition of the secret x to design a range proof in an interval of the form $[0, 2^k[$. Later, Schoenmakers [36] has proposed a more general solution which permits to prove that $x \in [a, b]$, using twice the method for range proofs in $[0, 2^k[$. Lipmaa *et al.* [28] have next presented a range proof for an interval $[0, b]$, based on the work from Damgård and Jurik [19]. This method uses the *multi-base decomposition* of the secret x in the *multi-base* b, which is a generalization of the above binary decomposition. From this initial multi-base decomposition solution [28], there are several ways to design a solution for a more general interval $[a, b]$. The first one is to use the Schoenmakers [36] technique which requires to use twice the range proof in $[0, b]$. The second one, considered in [28], exploits a generalization of the multi-based decomposition in order to directly obtain a characterization that x lies in an interval $[a, b]$. The result is very efficient since the time and space complexities are related to the size of the secret “plus one” (and not twice the size of the secret if one uses the Schnoenmakers’ technique).

More recently, Camenisch *et al.* proposed in [10] a new way to treat set membership and range proofs, based on an initial work from Teranishi and Sako [38]. This work has been refined by Chaabouni *et al.* in [14]. The main idea is that a designated authority produces public signatures on each element of the set

Φ (resp. interval $[a, b]$). The proof of knowledge of *e.g.* a secret $x \in \Phi$ (resp. $x \in [a, b]$) consists in proving the knowledge of a (public) signature on the secret x (which is possible only if x belongs to the interval) without revealing x nor the used signature. They next use the u-ary representation of the secret and the technique in [36] to improve the complexity of the final range proof, even if the number of signatures to be published should be equal to u.

From a practical point of view, it seems very hard to know which solution to choose since it may depend on the size of the secret, the size of the interval or the space (and time) complexity of the needed parameters (generation). Indeed, the size of the secret is important for the complexity of the second and third families of solutions, while they are not relevant for the square decomposition method. On the contrary, when a secret is small, the fact that one needs to use a group of unknown order for this latter method can be an important disadvantage.

As we are living in a world where more and more cryptographic computations are done inside a device with restricted computational capabilities, such as a smart card or a mobile phone, we think that:

1. it is essential to clearly know which method should be used, depending on the constraints, so as to optimize the obtained efficiency;
2. the improvement of these methods is very important to achieve maximum efficiency. As it seems difficult to improve the asymptotic complexities of these existing methods, it remains either to find new methods, or to improve existing ones at the constant factor level.

Regarding the first item, we propose in this paper the first complete study on all existing range proof methods. We thus compare the prover and verifier time complexities, the space complexity, the size of the public key and the time complexity of the setup phase. We then show that this is currently not possible to say that one solution is always the best one. A similar study has already been done in [10] but they do not consider the Lipmaa-Asokan-Niemi method [28], and they do not study, for example, the complexity from the verifier's side.

Regarding the second item, we first improve the Camenisch *et al.* signature based solution [10]. We also consider a new way to use the Lipmaa *et al.* [28] ideas, based on the multi-decomposition of the secret.

This paper is organized as follows. Section 2 gives the outlines of our study and introduces some useful tools. In Sections 3 and 4, we respectively present our new signature-based set membership and multi-base decomposition based range proofs. Before concluding, we give a complete efficiency analysis of all the existing range proofs, including our two new methods, in Section 5.

2 Notations and Useful Tools

Roughly speaking, a zero knowledge proof of knowledge (ZKPK) is an interactive protocol during which an entity proves to a verifier that he knows a set of secret values $\alpha_1, \dots, \alpha_\ell$ verifying a given relation $\mathcal{R}$ without revealing anything else. Such a proof is denoted $\textsc{Pok}(\alpha_1, \dots, \alpha_\ell : \mathcal{R}(\alpha_1, \dots, \alpha_\ell))$ in the following.

Let Φ be a public discrete set and $x \in \Phi$ the studied secret which is committed in Com. Our aim in this paper is to construct a non-interactive zero-knowledge proof of knowledge, using the Fiat-Shamir heuristic and the random oracle model [21,34], that the committed integer x lies in the set Φ. Such a proof is generally called "set membership proof". We also consider in this paper the special case of "range proof" where Φ consists in the range $\{a, a+1, \dots, b-1, b\}$, for $a, b \in \mathbb{N}$, denoted $[a, b]$.

As these proofs are related to an interactive version [21], our new methods are also relevant in this case. In fact, the main difference between the two cases is for the complexity study, where the size of the parameters (in particular the size of the challenges) can be different in each case.

We consider, except when explicitly mentioned, that we are working on an elliptic curve, and we use the multiplicative notation. More precisely, let $\mathbb{G}$ be a group corresponding to an elliptic curve $E/\mathbb{F}_p$ over $\mathbb{F}_p$, where p is a prime integer. Let q be a prime divisor of the group order and let E have embedding degree k with respect to q. The used commitment scheme is a Pedersen commitment [33]. The secret x is then committed as $Com = g^x h^r$, where $r \in_R \mathbb{Z}_q^*$.

2.1 Some Zero-Knowledge Proofs of Knowledge

In this paper, ZKPK are constructed over a cyclic group $\mathbb{G} = \langle g \rangle$ either of prime order q or of unknown order. A ZKPK should be complete (a valid prover is accepted with overwhelming probability), sound (a false prover should be rejected with overwhelming probability) and zero-knowledge (no information about the secret are revealed). We will use the following constructions: proof of knowledge of a discrete logarithm [35]: $\textsc{Pok}(\alpha : y = g^\alpha)$; proof of knowledge of a representation [32]: $\textsc{Pok}(\alpha_1, \dots, \alpha_q : y = g_1^{\alpha_1} \dots g_q^{\alpha_q})$; proof of equality of discrete logarithms [17]: $\textsc{Pok}(\alpha : y = g^\alpha \wedge z = h^\alpha)$; proof of the "or" statement [18] (proof of knowledge of one representation among k ones): $\textsc{Pok}(\{\alpha_{ij}; j \in J_i\} : \bigvee_{i=1}^{k} C_i = \prod_{j \in J_i} g_j^{\alpha_{ij}})$ and plaintext equivalence test [25].

From $Com = g^x h^r$, the predicate $(x = 1)$ (resp. $(x = 0)$) can be proven, in a zero-knowledge manner, by $\textsc{Pok}(r : Com/g = h^r)$ (resp. $\textsc{Pok}(r : Com = h^r)$). As a consequence, the predicate $(x = 1 \vee x = 0)$ is related to the proof of knowledge $\textsc{Pok}(r : Com/g = h^r \vee Com = h^r)$.

Remark 1. $\textsc{Pok}(r : Com/g = h^r)$ proves that $Com = gh^r$ and thus that $x = 1$. Consequently, this proof of knowledge is not zero-knowledge regarding x (but *w.r.t.* r). Nevertheless, $\textsc{Pok}(r : Com/g = h^r \vee Com = h^r)$ ensures this property as anybody is able to learn if x equals 0 or 1. The only information given by this proof is that x is a bit, which will be always publicly known in the following.

2.2 Boneh-Boyen Signatures with(out) Pairings

Boneh and Boyen have proposed in [7] a short signature scheme (BB for short), secure under the q-SDH assumption [7], for which it is possible to prove the knowledge of a signature on a message, without revealing the signature nor the message. On input a secret key y and a generator g of a group $\mathbb{G} = E/\mathbb{F}_p$ of prime order q, the signature of a message $m \in \mathbb{Z}_q$ is obtained by computing $\sigma = g^{1/(y+m)}$. Given a bilinear pairing e, a signature σ of m is valid if $e(\sigma, Yg^m) = e(g,g)$, where $Y = g^y$. The ZKPK of such a signature is $\textsc{PoK}(m, r, s : Com = g^m h^r \wedge e(T,Y) = e(T,g)^x e(g,g)^s)$, assuming that $T = \sigma^s$ is known to the verifier.

Our aim in the next section is to provide a set membership proof which does not ask the prover to perform pairing computations. This implies to use a signature scheme which (i) provides a ZKPK of a signature on a message, without revealing the message nor the signature and (ii) does not need a pairing computation to verify such signature.

Boneh-Boyen signature without pairings. Let $\mathbb{G}$ be a cyclic group with prime order q where the Decision Diffie-Hellman (DDH) problem is assumed to be hard and $\tilde{g}$ and g_1 two random generators of $\mathbb{G}$. The signer's private key is $y \in \mathbb{Z}_q^*$ and the corresponding public key is $Y = \tilde{g}^y$.

The signature on a message m is first the value $A = g_1^{\frac{1}{y+m}}$ computed as for the initial scheme. This implies that $A^y = g_1 A^{-m}$. Then since we work in a group $\mathbb{G}$ not equipped with a pairing, the signer has to additionally prove that the signature on m is valid, which is done by generating a ZKPK π that the discrete logarithm of $(g_1 A^{-m})$ in the base A is equal to the discrete logarithm of Y in the base $\tilde{g}$: $\textsc{PoK}(y : Y = \tilde{g}^y \wedge A^y = g_1 A^{-m})$. Finally, the signature on m is valid iff the proof π is valid.

Theorem 1. *The BB without pairing scheme is existentially unforgeable under a weak chosen message attack under the* $\mathfrak{q}$*-Strong Diffie-Hellman assumption, in the random oracle model.*

Proof (sketch). Under the $\mathfrak{q}$-Strong Diffie-Hellman assumption, and given a message m, it is impossible to find a A such that $A = g_1^{\frac{1}{y+m}}$ which is not given to the signing oracle, as proved in [7]. Moreover, in the random oracle model, the signature of knowledge π is unforgeable [21,34], which concludes the proof. □

2.3 A Threshold Cryptosystem

Our signature-based scheme relies on a threshold version of a semantically secure cryptosystem with homomorphic property, such as El Gamal [20]. Let $\mathbb{G}$ be a cyclic group of order q where the DDH problem is hard. The public key is composed of the elements $(\tilde{g}, z = \tilde{g}^x)$ with $\tilde{g}, z \in \mathbb{G}$ and the corresponding private key is formed by $x \in \mathbb{Z}_q$. The El Gamal ciphertext of a message $m \in \mathbb{G}$ is $(C_1 = \tilde{g}^r, C_2 = mz^r)$, where $r \in \mathbb{Z}_q$ is a random number. The decryption of the ciphertext (C_1, C_2) is obtained through the computation of $m = C_2/(C_1^x)$. The El Gamal cryptosystem is *semantically secure* under the DDH assumption. It is also homomorphic regarding the multiplication: $\textsc{Enc}(m_1) \cdot \textsc{Enc}(m_2) = \textsc{Enc}(m_1 \cdot m_2)$.

Moreover, in a threshold version [23], the El Gamal public key and its corresponding private key are cooperatively generated by n parties; though, the private key is shared among the parties. In order to decrypt a ciphertext, a minimal number of t out of n parties is necessary.

3 Our New Signature-Based Set Membership Proof

In this section, we present a new set membership protocol that bears some similarities with the protocol proposed by Camenisch *et al.* [10]. Yet, our set membership proof does not use pairings and is thus computationally more efficient, as we will see in Section 3.3. As a consequence, our proposal may be preferred whenever a range proof has to be implemented in *e.g.* a smart card which does not naturally implement a pairing. Finally, our solution implies that the verifier owns a decryption key and is consequently more relevant in the case of electronic voting [26,3] where the (set of) verifier(s) owns such key to open ballots.

3.1 Signature Based Characterization

The idea of using the signatures of all the integers in a public interval is due to Teranishi and Sako [38] and next used in [10] by Camenisch *et al.*. More recently, Chaabouni *et al.* [14] have proposed a more efficient variant in the case of a range proof only. The signature based characterization is the following one.

Lemma 1. *Let a, b, x be three integers. For all $k \in [a, b]$, let $\sigma_k = \textsc{Sign}(k)$ be a signature on the value k using the secret key of a designated authority. Let Σ be the set of all σ_k. Then, $x \in [a, b]$ if and only if $\exists \sigma \in \Sigma$ such that $\sigma = \textsc{Sign}(x)$.*

The authors of [38,10] propose to use BB short signatures [7] which, as said in Section 2.2, permits to prove the knowledge of a couple (message, signature) without revealing the message nor the signature. Camenisch *et al.* [10] refine this method by using the u-ary representation of the secret x. Then a proof that each digit of x in base u belongs to the interval $[0, u[$ is realized. This method requires to publish less signatures than the initial method. Then, they use [36] for the general interval $[a, b]$.

We introduce in the following a new signature-based range proof, based on the work in [10], which does not ask the prover to compute pairings. The basic idea is that the verifier $\mathcal{V}$ first sends to the prover a signature of every elements in the set Φ. Thus, the prover $\mathcal{P}$ picks a signature σ on the particular element s to which Com is a commitment. Then, he "encrypts" this signature and performs a proof of well-formedness. The verifiers (a threshold of them) then use their private keys to check whether the ciphertexts encrypt a valid signature or not (but without decrypting the corresponding ciphertexts). Note that two recent results [1,40] use a conceptually similar trick to make a proof of validity of a BB signature (yet, both still use pairings).

3.2 The Protocol in Details

Participants and Notation. Let $Com = g^x h^r$ be a commitment on x, where $g, h \in \mathbb{G} = E/\mathbb{F}_p$ and $r \in_R \mathbb{Z}_q$. The prover, denoted $\mathcal{P}$, picks a valid signature A on the committed element x. The verifiers are denoted as a set $\mathcal{V}$, *e.g.* the set of talliers in a voting scheme[1]. They share an El Gamal private key $\widehat{V}$ corresponding to a public key V. They also share the private key y associated to the BB signature without pairing public key Y. Finally, we denote $E_V[m]$ the El Gamal encryption of a message m computed with the public key V, and $D_{\widehat{V}}[m]$ the El Gamal decryption of m computed with the private key $\widehat{V}$.

Setup of the public parameters. This phase takes place before any set membership proof in order to establish the parameters, and especially the four generators $\tilde{g}$ $g_1, g_2, g_3 \in \mathbb{G}$. Then, the verifiers $\mathcal{V}$ collaborate to generate the encryption (resp. signature) public key V (resp. $Y = \tilde{g}^y$) and its corresponding private key $\widehat{V}$ (resp. y). The resulting key $\widehat{V}$ (resp. y) is not known by the verifiers individually. Each verifier $\mathcal{V}_i$ knows only a share v_i (resp. y_i) of this key. Their corresponding public key is denoted by $V_i = g^{v_i}$ (resp. $Y_i = \tilde{g}^{y_i}$).

After generating their keys, the verifiers are ready to issue the signatures of each element of the set $\varPhi$. The verifiers might generate the signatures in a threshold fashion by means of a scheme similar to [39].

The signature-based set membership protocol. After having computed a commitment Com on her secret x, the prover, denoted $\mathcal{P}$, picks a valid BB without pairing signature (A, π) on the particular element x, with $A = g_1^{\frac{1}{y+x}}$. She next selects a random $f \in \mathbb{Z}_q^*$ and computes $B = A^f$. Then, she generates the tuple $(B, E_V[B^{f^{-1}}], E_V[B^{xf^{-1}}], \varPi) = (B,\ E_V[A],\ E_V[A^x],\ \varPi)$ which is in the following denoted $= (B,\ C,\ D,\ \varPi)$. Within this tuple, the ciphertext $E_V[B^{f^{-1}}]$ is an encryption of the signature $A = g_1^{\frac{1}{y+x}}$ on the message x. The value $\varPi$ is a non-interactive zero-knowledge proofs which contains the following elements.

- ($\varPi_1$) A proof that $\mathcal{P}$ knows the plaintext related to the ciphertext $C = E_V[B^{f^{-1}}] = (C_1, C_2)$. In particular, $\mathcal{P}$ has to prove that she knows the representation of C_2 in the bases B and V using Okamoto's protocol [32].
- ($\varPi_2$) A proof that $\mathcal{P}$ knows the plaintext related to the ciphertext $D = E_V[B^{xf^{-1}}] = (C_3, C_4)$. Especially, $\mathcal{P}$ has to prove that she knows the representation of C_4 in the bases B and V using Okamoto's protocol [32].
- ($\varPi_3$) A proof that $\mathcal{P}$ knows the representation of Com in the base g and h.
- ($\varPi_4$) A proof that the discrete logarithm of C_4 in the base C_2 is equal to the discrete logarithm of Com in the base g using a variant of the discrete log equality test owing to Chaum and Pedersen [17].

In the end, $\varPi = \textsc{Pok}(\alpha, \beta, \theta, \delta, \lambda, \mu, \varepsilon : C = (\tilde{g}^\alpha, B^\beta V^\alpha) \wedge D = (\tilde{g}^\delta, B^\theta V^\delta) \wedge Com = g^\lambda h^\mu \wedge C_4 = C_2^\lambda V^\varepsilon)$. Finally, the prover sends $(B,\ C,\ D,\ \varPi)$ to the verifier, along with Com.

Verification. The verification phase corresponds to the following.

[1] In most e-voting systems, the voter has to prove that his choice belongs to the set of valid candidates [26].

1. $\mathcal{V}$ check that $B \neq 1$ and that all proofs in Π are valid and abort if one of these verifications fails.
2. $\mathcal{V}$ verify whether the tuple $(B, E_V[B^{f^{-1}}], E_V[B^{xf^{-1}}], \Pi)$ *encrypts* a valid signature A on a message x (i.e: satisfy the relation $A^{y+x} = g_1$) as follows.

   (a) $\mathcal{V}$ execute a Plaintext Equivalence Test [25] in order to check that C is not an encryption of 1. For this, $\mathcal{V}$ cooperatively select a random number $\alpha \in \mathbb{Z}_q^*$ and compute C^α. Then, they cooperatively decrypt C^α. If the decryption result is equal to 1, then C is an encryption of 1. Otherwise, the result will be a random number and this indicates that the encrypted plaintext is different from 1. $\mathcal{V}$ abort if this verification fails.

   (b) $\mathcal{V}$ next cooperatively compute $E_V[B^{f^{-1}}]^y = E_V[B^{yf^{-1}}]$ thanks to their secret keys y_i. Then, they use the El Gamal homomorphic property to obtain $E_V[A^{y+x}] = E_V[B^{yf^{-1}+xf^{-1}}] = E_V[B^{yf^{-1}}] \cdot E_V[B^{xf^{-1}}]$. Finally, from $E_V[B^{yf^{-1}+xf^{-1}}]$, the public parameter g_1 and the El Gamal homomorphic property, they deduce $E = E_V[B^{yf^{-1}+xf^{-1}} g_1^{-1}] = E_V[A^{y+x} g_1^{-1}]$.

   (c) In order to identify whether E encrypts a valid signature, $\mathcal{V}$ execute a Plaintext Equivalence Test [25]. Thus, they determine whether E is an encryption of the ciphertext 1 (valid signature) or not (invalid signature), as done before for C.

3.3 Comparison with Related Work

We now compare our new signature based set membership solution with the one in [10]. First, our scheme requires a cooperative verification between at least two actors. In the case of an interactive set membership proof, these two actors can be the verifier and the prover. We here argue that the prover will not decrypt the ciphertexts if she does not want to, thus the scheme is secure. In the case of e-voting, a committee of persons is generally responsible of the tally to avoid that single bullet can be decrypt. This committee might be used as well to perform the cooperative verification.

In Table 1, we use as a basic operation, for our efficiency comparison, the multiplication in the basic field[2] $\mathbb{F}_p$. In a nutshell, a multi-exponentiation with l terms, using the Shamir's trick, necessitates $|q|\frac{25}{3}\frac{2^{l+1}-1}{2^l}$ multiplications in $\mathbb{F}_p$. Using the results given in [2], a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ necessitates $(|q|-1)\big(3(k-1)k^2 + 2k + 23\big) = (|q|-1)\mathsf{P}(k)$ multiplications in $\mathbb{F}_p$, where k is the degree of the extension field and $\mathsf{P}(k) = 3(k-1)k^2 + 2k + 23$.

Our solution is then more efficient than Camenisch *et al.* one [10]. More precisely, with $|q| = 256$ and $k = 6$, the prover's time complexity in our case is better by a factor of 2. Moreover, the prover does not have to evaluate a pairing, which can ease the development of our solution in restricted devices. Regarding verifier's time complexity, our solution is better for a number of verifiers $l_v < 4$.

[2] Following [5], if one would like to use the multiplication in $\mathbb{G}$ as a basis for the comparison, one should have to remove a factor $\frac{25|q|}{2}$ from our complexities.

Table 1. Efficiency comparison of signature-based set membership proofs

Method	Prover's time complexity (multiplication in $\mathbb{F}_p$)	Verifier's time complexity (multiplication in $\mathbb{F}_p$)	Space complexity (in bits)	Size of public key (in bits)
Signature based [10]	$\big(\frac{125}{3} + \mathsf{P}(k)\big)\lvert q\rvert$ $+(36 - \mathsf{P}(k))$	$\big(\frac{1025}{24} + \mathsf{P}(k)\big)\lvert q\rvert$ $+\mathsf{P}(\mathrm{k})$	$2\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$ $+4q$	$(3 + \lvert\varPhi\rvert)\lvert\mathbb{G}\rvert$ $+\lvert\mathbb{G}_T\rvert$
Our method	$\frac{1975}{12}\lvert q\rvert + 6$	$\big(\frac{375}{2}\lvert q\rvert + 57\big)l_v$	$12\lvert\mathbb{G}\rvert + 8\lvert q\rvert$	$2l_v\lvert\mathbb{G}\rvert + 6\lvert\mathbb{G}\rvert$ $+3\lvert\varPhi\rvert\lvert\mathbb{G}\rvert + 2\lvert\varPhi\rvert q$

4 Our New Multi-base Decomposition Range Proof

In this section, we give our new method to prove that a secret value x belongs to a given public interval $[a, b]$, based on the *multi-base decomposition* of the secret x and the bound a in the *multi-base* b. We next compare a and x by using a generalization of a result due to Fischlin [22]. We also take advantage of the fact that both the prover and the verifier know the value a: if the most significant digits of a are equal to 1, then this is necessarily the case for the corresponding digits of every secret $x \in [a, b]$. Thus, without compromising the secrecy of x, we can reveal these digits by simply opening the corresponding commitments. We finally exploit some boolean logic results to obtain one of the most efficient range proof method for small secrets. The whole protocol for proving that a committed secret value x belongs to a given public interval $[a, b]$ is given in Section 4.5. The other intermediate protocols of this section are only given to better understand our ideas.

4.1 Multi-base Decomposition

In [28], Lipmaa *et al.* use the following characterization to prove that one secret is in $[0, v]$. We call this characterization the *multi-base decomposition.*

Lemma 2 (Multi-base decomposition). *Let v and x be positive integers and let $\ell = \lfloor \log_2 v \rfloor$. Then, $x \in [0, v]$ if and only if $x = \sum_{i=0}^{\ell} v_i x_i$ where for all $i \in [0, \ell]$, $x_i \in \{0, 1\}$ and $v_i = \lfloor (v + 2^{\ell-i})/2^{\ell-i+1} \rfloor$. In this case, we write $x = [\![x_0, \ldots, x_\ell]\!]_v$.*

As a direct consequence of this lemma, $v = [\![1, \ldots, 1]\!]_v$, which result can easily be proven by contradiction. This lemma can next be used to design a ZKPK that $x \in [0, v]$ by committing each x_i and next using the public v_i to prove the relation $x = \sum_{i=0}^{\ell} v_i x_i$ (see Section 4.2).

Uniqueness of the decomposition. For some value v, the decomposition of an integer x is not unique. For example, for $v = 10$, we have $v_3 = 5, v_2 = 3, v_1 = 1$ and $v_0 = 1$. Then, the integer $x = 6$ can be written either $[\![0, 1, 0, 1]\!]_v$ or

$[\![1,0,0,1]\!]_v$. However, our technique requires the uniqueness of the decomposition, at least for the public bounds. For this reason, we define a deterministic decomposition algorithm MBDec which outputs a unique decomposition for a given integer. This algorithm takes as input $(x, v_0, \ldots, v_\ell)$ and returns $x = [\![x_0, \ldots, x_\ell]\!]_v$. It first initializes $m = x$. Next, for $i = \ell$ to 0, if $m \geq v_i$, it states $x_i = 1$ and modifies $m := m - v_i$. Otherwise, it states $x_i := 0$ and m is not modified.

In the following, we always consider that this algorithm is used and, given a base v and an integer x, we thus assume the uniqueness of the multi-base decomposition of x in base v.

The Bit by Bit Characterization. The next characterization has been proposed by Fischlin [22] in the case of the binary representation. We here present it in the case of the multi-base decomposition of two different values. We compare these two values thanks to their representations.

Lemma 3 (Bit-by-bit Lemma). *Let v, $a = [\![a_0, \cdots, a_\ell]\!]_v$ and $x = [\![x_0, \cdots, x_\ell]\!]_v$ be three positive integers. Then, $a < x$ if and only if $\exists i' \in [0, \ell]/a_{i'} = 0$, $x_{i'} = 1$ and $\forall j > i', a_j = x_j$.*

4.2 Focus on $x \leq b$

We first focus on the case $x \leq b$. This part of the proof is close to the ones proposed in [19,28] which treat the case of a group of unknown order (see also Remark 2 below).

The idea is to decompose the secret in a multi-base while considering the case $v = b$ (we will stay in that case for the rest of the paper). Indeed, if we denote $\ell = \lfloor \log_2 b \rfloor$, the proof that $x \leq b$ is obtained using that $x = [\![x_0, \ldots, x_\ell]\!]_b$. We commit each x_i and prove that (i) each $x_i \in \{0, 1\}$ and that (ii) the relation $x = \sum_{i=0}^{\ell} b_i x_i$ holds. We here remember that each $b_i = \lfloor (b + 2^{\ell - i})/2^{\ell - i + 1} \rfloor$ can be computed by both the prover and the verifier, since b is publicly known. The proof that $x \in [0, b]$ is described as follows.

1. For all $i \in [0, \ell]$, the prover randomly chooses $r_i \in_R \mathbb{Z}_q$ and sends $C_i = g^{x_i} h^{r_i}$ to the verifier.
2. Both the prover and the verifier can compute $\tilde{C} = \prod_{i=0}^{\ell} C_i^{b_i}$, which is equal to $g^x h^{\sum_{i=0}^{\ell} b_i r_i}$ iff the x_i's correspond to the multi-base decomposition of x in base b. Let $t = \sum_{i=0}^{\ell} b_i r_i$.
3. Finally, the prover and the verifier play the following[3] ZKPK

$$U_1 = \textsc{PoK}\big(x, t, r_0, \ldots, r_\ell : (C_0 = h^{r_0} \vee C_0/g = h^{r_0}) \wedge \ldots \\ \wedge (C_\ell = h^{r_\ell} \vee C_\ell/g = h^{r_\ell}) \wedge \tilde{C} = g^x h^t\big)$$

Remark 2. In fact, one may think that we only prove that $x = \sum_{i=0}^{\ell} b_i x_i \pmod{q}$. But, as the x_i's belong to $\{0, 1\}$, $\sum_{i=0}^{\ell} b_i x_i \leq \sum_{i=0}^{\ell} b_i = b < q$, since

[3] The second part of U_1, related to $\tilde{C}$, proves that the relation $x = \sum_{i=0}^{\ell} b_i x_i$ holds.

we consider, for obvious reasons, that $2^\ell < q$. Thus, $x = \sum_{i=0}^{\ell} b_i x_i$, in $\mathbb{Z}$. So, this method is very efficient in a group of prime order but not so efficient in the case of group of unknown order [19,28]. In this case, it remains to prove that the equality holds in $\mathbb{Z}$, *e.g.* by using a proof of knowledge of three integers α, β, γ such that $\alpha = \beta\gamma$, using as an example [19].

4.3 Treatment of the Most Significant Digits of $a \leq x$

In the above method, we proved that the secret x can be represented in the multi-base b. Due to Lemma 2, this permits to prove that $x \leq b$. We now focus on the case $x \geq a$, considering the above method. We also represent a in the multi-base b, that is, we have $a = [\![a_0, \dots, a_\ell]\!]_b$, since $a < b$. We now compare x and a using their respective multi-base decompositions in base b, which permits some simplifications.

As both the prover and the verifier know the value a, and thus its representation in the multi-base b, we can use a trick based on the following result.

Lemma 4. $\exists! i_0 \in [0, \ell] / a_{i_0} = 0 \wedge \forall i > i_0, a_i = 1.$

Proof. The existence is given by the fact that $a \neq b$ and the uniqueness is verified assuming that the MBDec algorithm of Section 4.1 is used[4]. □

This lemma says that some of the most significant digits of a, in multi-base b, may be equal to 1. It also says that the first 0-bit of a is denoted i_0. Thus, using this lemma, and considering that both $b_i = a_i = 1$ for all $i \in [i_0 + 1, \ell]$ (see Section 4.1 for the b_i's), we obtain that if $x \in [a, b]$ then $\forall i \in [i_0 + 1, \ell], x_i = 1$.

The verifier knows a and b, thus the prover does not reveal any secret information if she simply opens the commitment on the x_i, *i.e.* she reveals the corresponding r_i. This trick permits to save the corresponding part of the predicate U_1 (*i.e.* $\bigwedge_{j=i_0+1}^{\ell} \big((x_j = 0) \vee (x_j = 1)\big)$). Thus, instead of proving $x \geq a$, the verifier and the prover should determine i_0, and only focusing on the proof that $\tilde{x} \geq \tilde{a}$, where $\tilde{a} = [\![a_0, \dots, a_{i_0}]\!]_{\tilde{b}}$ and $\tilde{x} = [\![x_0, \dots, x_{i_0}]\!]_{\tilde{b}}$, with $\tilde{b} = \sum_{i=0}^{i_0} b_i$.

Note that i_0 only depends on a and b. Thus, it should be computed at the creation of a and b, and may be considered as a public parameter.

4.4 Remaining Digits of $a \leq x$

We now focus on the comparison between the values $\tilde{a} = [\![a_0, \dots, a_{i_0}]\!]_{\tilde{b}}$ and $\tilde{x} = [\![x_0, \dots, x_{i_0}]\!]_{\tilde{b}}$. For this purpose, we use the bit-by-bit Lemma 3, which says that $a < x$ if and only if $\exists i' \in [0, \ell] / a_{i'} = 0$, $x_{i'} = 1$ and $\forall j > i', a_j = x_j$.

Study of the i-th digit. Under the fact that the verifier already knows a, we thus have the following for the i-th digit of a.

[4] This is always the case since the decomposition of a is done at the generation phase by some trusted authorities.

- If $a_i = 0$, there are two cases. If $x_i = 0$, we have to prove it and compare the remaining digits of $\tilde{a}$ and $\tilde{x}$, using the same method. If $x_i = 1$, we have to prove it, which is enough to claim that $a \leq x$.

  As we want to prove in a zero-knowledge manner that $a \leq x$, we have to consider both cases ($x_i = 0$ or 1), without revealing which statement is true. For this purpose, we use an "or" statement. If we denote by B the statement regarding the comparison between the remaining digits of a and x (in the case $x_i = 0$), this gives the predicate $(x_i = 1) \vee \big((x_i = 0) \wedge B\big)$.

- If $a_i = 1$, by construction of the recursion, we necessarily have $x_i = 1$. Then, we have to prove it and to compare the remaining digits of $\tilde{a}$ and $\tilde{x}$ (from 0 to $i - 1$), using the same method. This gives the predicate $(x_i = 1) \wedge B$.

As the verifier knows the value a, she is able to decide whether the prover has to perform the first type of proof or the second one. So we do not need to use another "or" statement for both cases ($a_i = 0$ and $a_i = 1$).

The special case $i' = i_0$. By definition, $a_{i_0} = 0$. We consequently necessary fall in the first case, and we must prove that $(x_{i_0} = 1) \vee \big((x_{i_0} = 0) \wedge B\big)$, with B the statement regarding the comparison between the i_0 less significant digits of a and x. We now remark that if this predicate is true, the verifier is convinced that x_{i_0} is a bit. Consequently, the prover has not to prove that $(x_{i_0} = 0 \vee x_{i_0} = 1)$ in the first part of the proof, which slightly reduces the proof's size.

Simplification of the first case. Regarding the first case ($a_i = 0$), the predicate $(x_i = 1) \vee \big((x_i = 0) \wedge B\big)$ can be further simplified. If we denote by A the predicate $(x_i = 1)$, then the predicate $(x_i = 0)$ is $\neg A$ (as it is proven in the proof U_1, given above, that x_i is a bit). Moreover, as B is related to the remaining digits of a and b, it does not contain any predicate on x_i. Thus the predicates A and B are independent. Finally, in our main proof, the predicate B is never used again, while A is connected to the studied predicate by an "and" statement between $A \vee (\neg A \wedge B)$ and $A \vee \neg A$. As $A \vee \neg A$ is necessary true, we obviously have $A \vee (\neg A \wedge B) = A \vee B$.

Description of the main algorithm. Using these results, we can now define the FGreat algorithm which, on input $\{a_0, \cdots, a_i\}$, outputs the logical relation L which needs to be used to prove that $x \geq a$. For this purpose, it first states that $L := \emptyset$ if $(a_i = \ldots = a_0 = 0)$. Next, if $(a_i = 1)$ there are two cases: either $(i = 0)$ and it defines $L := (x_0 = 1)$ or it states that $L := (x_i = 1) \wedge [\text{FGreat}(\{a_0, \cdots, a_{i-1}\})]$ (which needs a recursive execution of FGreat). Finally, if $(a_i = 1)$ then $L := (x_i = 1) \vee [\text{FGreat}(\{a_0, \cdots, a_{i-1}\})]$.

In consequence, the prover should prove that the following predicate is true:

$$(x_{i_0} = 1) \vee [(x_{i_0} = 0) \wedge \text{FGreat}(\{a_0, \cdots, a_{i_0 - 1}\})] \, .$$

4.5 Our Range Proof Protocol

Let b, $a = [\![a_0, \cdots, a_\ell]\!]_b$ and $x = [\![x_0, \cdots, x_\ell]\!]_b$ be three integers such that $a < b$ and $x \in [a, b]$. At the key generation phase, one has to first determine i_0 such that $a_{i_0} = 0$ and $\forall i > i_0, a_i = 1$. It next executes the FGreat algorithm on input $\{a_0, \cdots, a_{i_0-1}\}$ in order to obtain L. During the main protocol, the prover and the verifier follow the following steps:

1. for all $i \in [0, \ell]$, the prover randomly chooses $r_i \in_R \mathbb{Z}_q$ and sends $C_i = g^{x_i} h^{r_i}$ to the verifier;
2. both the prover and the verifier can compute $\tilde{C} = \prod_{i=0}^{\ell} C_i^{b_i}$. Let $t = \sum_{i=0}^{\ell} b_i r_i$;
3. for all $i \in [i_0 + 1, \ell]$, the prover reveals r_i;
4. the prover and the verifier then play the following interactive ZKPK

$$U_f = \text{Pok}\Big(x, t, r_0, \ldots, r_{i_0} : \tilde{C} = g^x h^t \wedge (C_0 = h^{r_0} \vee C_0/g = h^{r_0}) \wedge \ldots \wedge \\ (C_{i_0-1} = h^{r_{i_0-1}} \vee C_{i_0-1}/g = h^{r_{i_0-1}}) \wedge \big(C_{i_0}/g = h^{r_{i_0}} \vee (C_{i_0} = h^{r_{i_0}} \wedge L)\big)\Big).$$

5 Efficiency Comparison of Range Proof Methods

In this section, we compare the efficiency of both related work and our proposals to decide which solution has to be used in which cases. Note that each proof considers that the commitment on x has already been done and thus, this step is not considered here. We also consider the setting given in Section 2.

We resume our main results in Table 2, in which we consider a security level of 128, which corresponds to the following values: $|q| = 256$, $|\mathbb{G}| = 257$, $|\mathbb{G}_T| = 1542$, $l_{\mathbb{Z}_n} = 3248$, $l_e = l_s = 160$. Due to space limitation, the related complexities will be detailed in the extended version of the paper (see Appendix A for some preliminary materials). The value ℓ corresponds to the size of the secret in bits.

Note that [14] does not appear in our comparison as its complexity is really close to those of [10]. The more recent proposal [15] does not appear in this comparison since its main contribution is to obtain a construction in the standard model (and not in an idealized model as it is our case), but at the detriment of the efficiency. Note finally that the proposal in [41] is broken (see [14]). We also give some graphics, for which we use the captions given in Figure 1 (left part).

Verifier's Complexity Comparison. The verifier's efficiency comparison between existing methods is given in Figure 2 (on the left). For $\ell > 25$, it is clear that signature-based solutions are the most interesting ones. When ℓ is small, $\ell \leq 5$, then our new multi-based decomposition solution become the best one, while the initial work from [28] is ideal when $5 < \ell \leq 25$.

Prover's Complexity Comparison. The prover's efficiency comparison between all methods is given in Figure 2 (on the right). Results are relatively similar to the verifier's ones, except for the breaking points. This time, signature based solutions are the most interesting ones when $\ell > 60$, while ours is ideal

Table 2. Time and space efficiency comparison

Method	Prover's time complexity (mod mul in $\mathbb{G}$)	Verifier's time complexity (mod mul in $\mathbb{G}$)	Space complexity (in bits)	Size of public key (in bits)
Binary [36]	$12503\ell + 30636$	$14935\ell + 24532$ $-2133 \times 2^{-\ell}$	$3592\ell + 2568$	2052
Multi-base [28]	$9600\ell + 20800$	$7468\ell + 23201$ $+533 \times 2^{-\ell}$	$1795\ell + 2565$	$256\ell + 1794$
Square [27]	$322\ell + 1.7 \times 10^7$	$322\ell + 9.87 \times 10^6$	$2.5\ell + 40474$	4082
Square [24]	$1272\ell + 5.44 \times 10^6$	$644\ell + 5.68 \times 10^6$	$4\ell + 24647$	5367
Signature[5] [10]	$17075k + 318852$	$1181k + 3210 \times 2^{\ell/k}$ $+148302$	$257 \times 2^{\ell/k}$ $+4365k + 6160$	$256k + 1796$
This paper	$467\ell^2 + 14068\ell$ $+5467$	$11202\ell + 2666$ $+1066 \times 2^{-\ell}$	$2564\ell - 768$	$\ell + 770$

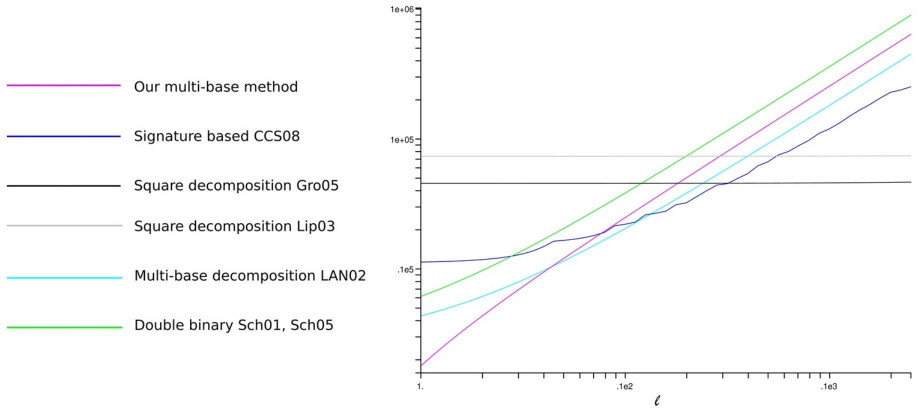

Fig. 1. Caption for graphic (on the left) and space's (on the right) efficiency comparison for different values of ℓ

when $\ell \leq 3$. Again, the initial multi-base decomposition scheme [28] is the best one when $3 < \ell \leq 60$.

Space's Complexity Comparison. We finally compare the space complexities in Figure 1 (right part), with the conclusion that, again, our method is very interesting when $\ell \leq 5$. Next, for $5 < \ell \leq 14$, the double binary method [36,37] is the most interesting one, while for $14 < \ell \leq 24$, one has to choose a signature-based method. Finally, for $\ell > 24$ the square decomposition methods [27,24] are the most efficient ones.

[5] The range proof $x \in [a, b]$ is done by using the u-ary representation of x and the parameter k is such that $u^k < b < u^{k+1}$.

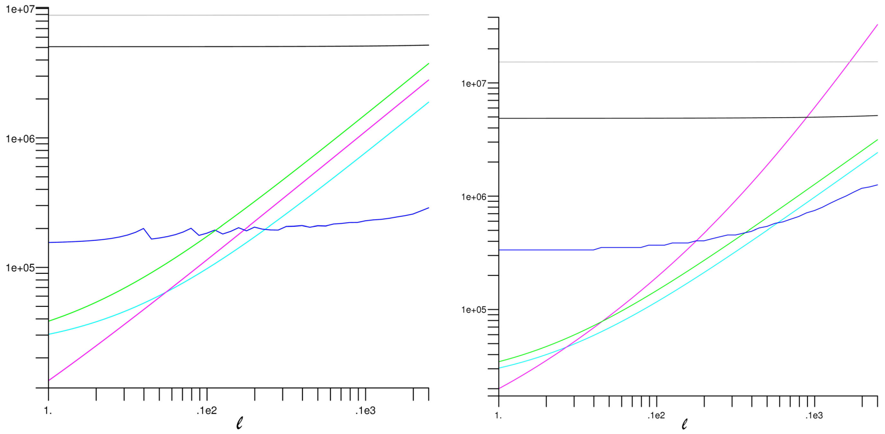

Fig. 2. Verifier's (on the left) and prover's (on the right) efficiency comparison for different values of ℓ

Conclusions on the Efficiency. We proved that several existing range proof methods are useful in practice. Indeed, it seems that for $\ell \leq 3$, our new multi-base decomposition method is the one to be chosen, while the Lipmaa-Asokan-Niemi solution [28] is the best one for $5 < \ell \leq 25$ and signature-based methods are more interesting when $\ell > 60$. Then, for $3 < \ell \leq 5$ and $25 < \ell < 60$, the choice should be done in accordance to the specificities of the system, regarding the computational strength of both the prover and the verifier, and the flow between them. Note that in some particular cases, our method becomes more interesting. In fact, this strongly depend on the values a and b.

Acknowledgments. The work of the first and fourth authors has been partially supported by the French ANR-11-INS-0013 LYRICS Project. We are grateful to Céline Dulong for helpful discussions on the multi-base decomposition version, and to anonymous referees for their valuable comments.

References

1. Acar, T., Chow, S.S.M., Nguyen, L.: Accumulators and U-prove revocation. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 189–196. Springer, Heidelberg (2013)
2. Arene, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster computation of the tate pairing. Cryptology ePrint Archive, Report 2009/155 (2009), `http://eprint.iacr.org/`
3. Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: PODC, pp. 274–283 (2001)
4. Bellare, M., Goldwasser, S.: Verifiable partial key escrow. In: ACM Conference on Computer and Communications Security, pp. 78–91 (1997)
5. Bernstein, D.J., Lange, T.: Explicit-formulas database. In: EFD (2009), `http://www.hyperelliptic.org/EFD/`

6. Blake, I., Seroussi, G., Smart, N.: Elliptic curves in cryptography (1999)
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
8. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
9. Brickell, E.F., Chaum, D., Damgård, I.B., van de Graaf, J.: Gradual and verifiable release of a secret. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 156–166. Springer, Heidelberg (1988)
10. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
11. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
12. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
13. Canard, S., Gouget, A., Hufschmitt, E.: A handy multi-coupon system. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 66–81. Springer, Heidelberg (2006)
14. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive combinatorics and discrete logarithm based range protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg (2010)
15. Chaabouni, R., Lipmaa, H., Zhang, B.: A non-interactive range proof with constant communication. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 179–199. Springer, Heidelberg (2012)
16. Chan, A., Frankel, Y., Tsiounis, Y.: Easy come - easy go divisible cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
17. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
18. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
19. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
20. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
21. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
22. Fischlin, M.: A cost-effective pay-per-multiplication comparison method for millionaires. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 457–472. Springer, Heidelberg (2001)
23. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999)
24. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)

25. Jakobsson, M., Juels, A.: Addition of elgamal plaintexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 346–358. Springer, Heidelberg (2000)
26. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES 2005, pp. 61–70. ACM (2005)
27. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
28. Lipmaa, H., Asokan, N., Niemi, V.: Secure vickrey auctions without threshold trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer, Heidelberg (2003)
29. Lynn, B.: On the Implementation of Pairing-based Cryptosystems. PhD thesis, Stanford University (2007)
30. Möller, B.: Algorithms for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
31. European Network of Excellence in Cryptology II. Ecrypt2 yearly report on algorithms and keysizes (2008-2009) (2009)
32. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
33. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
34. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13, 361–396 (2000)
35. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
36. Schoenmakers, B.: Some efficient zero-knowledge proof techniques. In: Workshop on Cryptographic Protocols (2001)
37. Schoenmakers, B.: Interval proofs revisited. In: Workshop on Frontiers in Electronic Elections (2005)
38. Teranishi, I., Sako, K.: k-times anonymous authentication with a constant proving cost. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 525–542. Springer, Heidelberg (2006)
39. Wang, H., Zhang, Y., Feng, D.: Short threshold signature schemes without random oracles. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 297–310. Springer, Heidelberg (2005)
40. Yang, Y., Ding, X., Lu, H., Weng, J.: Self-blindable credential: Towards lightweight anonymous entity authentication. Cryptology ePrint Archive, Report 2013/207 (2013), `http://eprint.iacr.org/`
41. Yuen, T.H., Huang, Q., Mu, Y., Susilo, W., Wong, D.S., Yang, G.: Efficient non-interactive range proof. In: Ngo, H.Q. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 138–147. Springer, Heidelberg (2009)

A Complexity Tools

As we are considering that we work over an elliptic curve, the basic operation we will use is the addition of points in the curve. Using ECRYPT II Recommendations [31] with a security level of 128, the elements of $\mathbb{F}_p$ are typically 256-bits integers. Thus, as it is necessary to store a point using the x-coordinate plus additional one bit to know which y-coordinate is to be chosen [6], a point needs 257 bits to be stored. The integer q should also be chosen as a 256-bit integer.

Modular Multiplication in $\mathbb{Z}_n$. As we are considering a security level of 128, we need to use a 3248-bits RSA modulus, according to [31]. Using the bouncycastle Java implementation of modular multiplication and the point addition in the secp-256r1 elliptic curve, we obtain that $M_{3248} = 5.2A_{256}$ where M_{3248} is the cost for the modular multiplication and A_{256} is the cost for the points addition. Moreover, a point addition in an Edwards curve [5] (in our case a multiplication in the group $\mathbb{G}$) necessitates 12 multiplications in the base field $\mathbb{F}_p$.

Use of Shamir's Trick. We also recall that the computation of a representation $c = \prod_{i=1}^{l} g_i^{e_i}$ can be improved by the use of the well-known Shamir's trick, presented in [30]. Note that this technique is generic and consequently also work for the elliptic curve case. In a nutshell, it is not necessary to compute each modular exponentiation and multiply the results since c can be computed globally. In most cases, this permits to save lots of computation. We thus consider that the computation of c necessitates approximatively $\frac{2^{l+1}-1}{3\times 2^{l-1}}$ times the cost of a modular exponentiation modulo a q-bits integer. Note moreover that this is a well-known result in the modular arithmetic theory that a modular exponentiation with an exponent of size e modulo a n-bits number corresponds to $\frac{3}{2}e$ modular multiplications modulo a n-bits number. We can conclude that one multi-exponentiation with l terms necessitates $b\frac{2^{l+1}-1}{2^l}$ modular multiplications, where b is the greatest bit length of the e_i's. Regarding elliptic curves, if we consider one scalar multiplication with a scalar of size e, then we have $S_{256} = \frac{25e}{2}\mathbf{m}$, using the results from [5] for Edwards curves, where S_{256} is the cost for the scalar multiplication (equivalent to a modular exponentiation) and $\mathbf{m}$ is the cost of multiplication in the base field $\mathbb{F}_p$. This gives $b\frac{25}{3}\frac{2^{l+1}-1}{2^l}\mathbf{m}$ for a multi-scalar multiplication (equivalent to a multi-exponentiation) with l terms.

The Case of Pairings. It is today not an easy task to obtain the efficiency of a pairing evaluation which can be used in our purpose. We have made the choice of using the recent work from Arène, Lange, Naehrig and Ritzenthaler [2] on the Tate pairing for Edwards curves, which is, to the best of our knowledge, the most efficient pairing implementation. It is written in this paper that the evaluation of the reduced Tate pairing, given by $e : E(\mathbb{F}_p)[q] \times E(\mathbb{F}_{p^k})/qE(\mathbb{F}_{p^k}) \rightarrow \mu_q$ where $\mu_q \subset \mathbb{F}_{p^k}^*$ denotes the group of q-th roots of unity, needs $(|q|-1)$ iterations of one point doubling and one point addition. If $\mathbf{s}$ denotes the cost of squaring in the base field $\mathbb{F}_p$ and if $\mathbf{M}$ and $\mathbf{S}$ denote the costs of multiplication and squaring in the extension field of degree $k = 6$, then the doubling step takes $1\mathbf{M} + 1\mathbf{S} + (k + 6)\mathbf{m} + 5\mathbf{s}$ while the mixed addition step needs $1\mathbf{M} + (k + 12)\mathbf{m}$. Assuming that $\mathbf{M} = \mathbf{S}$ and $\mathbf{m} = \mathbf{s}$, and, using [29], that $\mathbf{M} = (k-1)k^2\mathbf{m}$, we obtain that one pairing evaluation P_{256} necessitates $(|q|-1)(3(k-1)k^2 + 2k + 23) = 575(|q|-1)$, for $k = 6$, multiplications in the base field. In the following, we consider the pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ where $\mathbb{G}_1$ necessitates 256-bits elements, with an order q of size 256 and $\mathbb{G}_T$ needs to manipulate elements of size 1542 (with $k = 6$). Note that using the above results, we obtain that $P_{256} \approx 46S_{256}$.

STUNT: A Simple, Transparent, User-Centered Network of Trust

Klaus Potzmader, Johannes Winter, and Daniel Hein

Institute for Applied Information Processing and Communications,
Inffeldgasse 16a,
8010 Graz,
Austria
klaus.potzmader@student.tugraz.at,
{johannes.winter,daniel.hein}@iaik.tugraz.at

Abstract. Secure end-to-end communication requires endpoint authenticity. Authenticating an endpoint in large networks, that is assuring that the other communication party is indeed who he or she claims to be, is a non-trivial task. Currently, the adopted solution is to rely on trusted third parties, who vouch for a certain host's authenticity. Recent incidents at renowned trusted third parties, as well as long standing problems, indicate a need for alternative solutions. We propose STUNT, a system that helps users to assess a host's authenticity by its trust relationships with other hosts. Hosts operated by service providers have to establish mutual trust relationships with other service providers to appear trustworthy to a user. These trust relationships are both limited and expensive, and thus STUNT enforces careful trust decisions by service operators. Clients are able to verify these trust relationships by cryptographic means. The verified trust relationships are presented to the users, to assist them with assessing the authenticity of the host. Ultimately, the trust decision rests with the user, leading to an individual, self-maintained trust base. We believe that, given the right tools, people are very well able to decide on a host's authenticity, and describe a possible technical concept to support informed decision-making.

Keywords: network, internet, SSL, authenticity, trust, proof of work.

1 Introduction

Secure communication over an untrusted network requires endpoint authentication. Currently, the most widespread approach to solving authenticity in a large, heterogeneous network is using trusted third parties to vouch for the genuineness of a host. These trusted third parties reside at the top of a hierarchic structure, where upper-level entities vouch for the authenticity of entities below, as described in the ITU-T X.509 [17] standard. The top-level trust entities are root certificates, issued by trusted third parties, called certificate authorities, and have to be trusted by clients for any subordiniate authority or leaf certificate to be considered trustworthy. Nowadays, software using this system

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 65–82, 2014.

typically comes with a predefined list of such trustworthy roots. According to the Electronic Frontier Foundation's SSL Observatory[1], about 650 such certificate authorities exist. The Microsoft Windows SSL root certificate member program lists 353 trusted authorities as of December 2012 [22]. The question is, however, why all users should trust this many authorities by default, without any prior knowledge about or personal contact with them at all.

Criticism about X.509 was ever-present, but got stirred up to new heights after recent incidents, such as the successful break-in into the Dutch certificate authority DigiNotar [16]. The ability to break into established certificate authorities and thus create valid certificates for any desired domain gives attackers incredible power. Such an attacker may be able to successfully mount Man-in-the-Middle attacks, without users recognizing the attack, even though a seemingly secure connection is used.

There are design issues that render the established system vulnerable as well. The fact that each certificate authority can issue certificates for any arbitrary domain, even ones that already got a certificate from another CA, poses another problem. As Sotirov and Zusman put it once, it ends up in a "*race to the bottom*"[2] scenario, where a CAs security principles end up to be only as good as the weakest principles of a CA present in the predefined trusted authorities of the client. In addition to early alternatives such as KeyNote and PolicyMaker [6–8], there are several recently proposed mitigation attempts for this, such as CertLock [24], Public Key Pinning [13], TACK [21] or Certificate Transparency [20], the latter three of which are Internet Engineering Task Force (IETF) drafts currently under review. Nevertheless, even without considering problems such as the complexity of proper revocation as well as certificate and certificate chain validation [14,15], trust in the currently employed system seemed to deteriorate during the last couple of years [2,11,12].

Therefore, we think that it is worth trying to find new solutions, which no longer involve a third party to decide about what host is or is not trustworthy. We do not think that these problems discussed above can be remediated without proper user integration into the trust decision process. For example, the current PKIX-based process hides all details of the trust decision from the user. This approach is fragile, because if the automatic process fails, and the unaware and unprepared user is called to intervene, there is a high probability that the user is overstrained and will thus simply ignore the problem [5,25,26].

Putting regular users into the position of making an informed decision is the crucial point of such a system, and we strive to create a recommendation-based system that works with a concept that is already familiar from everyday life. We propose STUNT, a system which enables the assessment of a host's authenticity for users by showing the surrounding network of trusting hosts. The system is designed to be decentralized, without the need for a trusted third party.

[1] `http://www.eff.org/observatory`

[2] In their BlackHat 2009 talk called *Breaking the Security Myths of Extended Validation SSL Certificates*, `http://www.blackhat.com/presentations/bh-usa-09/ZUSMAN/BHUSA09-Zusman-AttackExtSSL-SLIDES.pdf`

The trust network is built in a way that cryptographically ensures that the shown relationships are indeed meaningful and cannot be falsely claimed. STUNT is based on the following design principles:

User-central. Users are the ones who decide whether a host is trustworthy or not, and there exist individual trust bases per user or software component.

Trust Relationships. Trust relationships are an expression of confidence by an entity, that the recipient is indeed what is claimed. The authenticity of a host is assessed by its trust relationships. Users decide whether a certain cluster of trusting hosts is reasonable or not. Trust relationships from users to servers are unidirectional. Users trust hosts, not vice versa. Between servers on the other hand, only mutual trust relationships exist.

Expensive Creation. Trust relationships between servers are intentionally expensive to create, as incentive for investing only in long-living relationships. Furthermore, the number of trust relationships is limited, such that the individual weight of a trust relationship is higher. Operators are thus encouraged to choose their trustees carefully.

Cheap Verification. Users are able to verify the presented relationships on the fly.

Risk-based Revocation Support. The system uses a whitelisting approach, where no host is considered valid, except all of its trust relationships verify correctly and are current. An active revocation scheme is embedded, which requires continuous updates in predefined, risk-based intervals for a relationship to be kept alive.

The Backfiring Principle. As far as possible, all attempts to misbehave shall backfire at the initiator.

To allow for informed user decisions, the surrounding trust relationships are presented to the user. Each node in that trust network represents another host whose operators decided to trust the host in question. Therefore, exploring a trust network is basically equivalent to checking references before trusting someone. This is a concept most people will be familiar with.

The system is built on the assumption that users already have at least a vague idea about the host in question and its trust relationships when confronted with the decision of trustworthiness. For example, if a user's online banking site is linked to other renowned institutes, this might be an indication that the host is indeed trustworthy. If, however, one ends up assessing a host's trustworthiness and is presented with a network of only unknown hosts, the system is bluntly telling the user that nothing about the host in question is known. The user is then free to decide whether to browse the presented network deeper, until she is able to make an informed decision. This decision could be to leave the host, to not enter any account information, or to trust this host anyways. The latter choice is risky and the corresponding risk has to be communicated via the user interface. This is similar to known warnings about self-signed certificates and similar occurrences.

We strive to avoid the problems outlined above by both a decentralized structure and user-based decision making. No entity is powerful enough to decide for someone else, and the influence gained by compromising an entity is limited to its direct neighbors. Moreover, we expect this approach to be intuitively usable, as it strongly resembles social interaction and everyday trust evaluations.

2 Related Work

The system is, to the best of our knowledge, a new approach to host authenticity on large networks. However, it can be loosely compared to systems such as Perspectives [28]. In Perspectives, a user can define a couple of notaries whom she trusts and queries them for the certificate fingerprint of a host in question. The host's authenticity is thereby dependent on whether all notaries acknowledge that fingerprint. Since the idea is to distribute notaries all around the world, it gets very hard for an adversary to fake these responses. The Perspectives prototype has been adopted in Convergence [27], a follow-up which aims at becoming a ready-to-use implementation of the Perspectives idea.

Furthermore, the system is loosely inspired by PGP[3], although with completely different internal workings. PGP supports transitivity using so-called trusted introducers, which are basically the equivalent of certificate authorities in an X.509 [17] world. There is no such counterpart in STUNT. Besides, there is a clear distinction between users and service providers, for whom very different rules apply.

In social and semantic web contexts, as well as in the context of autonomous agents, many reputation-based trust systems have been proposed [3, 18]. Reputation-based systems share the approach of determining a user's or host's trustworthiness by the people or systems who vouch for the instance in question. We pursue a similar approach in the proposed system. Contrary to autonomous systems, however, the users and service operators are an integral part of the outlined system.

Web of Trust[4] introduced a similar system for rating web pages on the Internet. To assert the safety of a web site, judged by a few predefined and individually rated categories, users rate websites regarding a predefined set of categories. Other users get aggregated votings as a guideline about the visited page's safety. STUNT is also based on a recommendation approach, but has a very different reference system in place. Contrary to Web of Trust, the recommendations are not voting-based. Instead, we use individual trust relationships for hosts to appear trustworthy. Furthermore, the reference mechanism only allows mutual trust relationships to ensure careful assertions. STUNT is best comparable to checking recommendations one gets from partner companies or reference clients before contracting with a previously unknown company.

3 System Design

The system design will be discussed in detail throughout this section, following a brief overview about the underlying concepts. In STUNT, users trust hosts in

[3] http://www.pgpi.org

[4] http://www.mywot.com

a unidirectional way, while the publicly visible trust relationships between hosts are mutual. The mutuality acts as an incentive to actually engage in trusts, as it is a time-consuming, expensive and, to some extent, risky task. Figure 1 shows such a sample trust network, where the relationships between hosts are publicly visible and user-to-host trust is not.

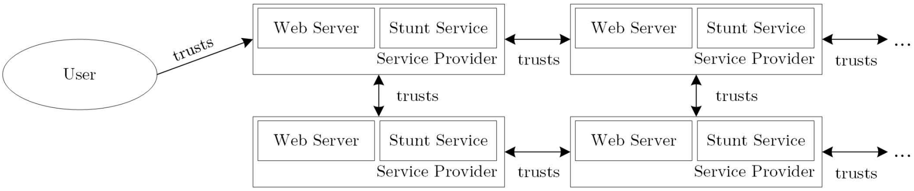

Fig. 1. Schematic of user trust and between-host trust

The system is based on the cornerstones of *user centrality*, *mutual and unidirectional trusts*, *expensive relationship creation*, *cheap relationship verification*, *risk-based revocation support* and what we call the *backfiring principle*. Expensive relationship creation is enforced by requiring both parties to compute what *BitCoin* [23] calls a *proof of work*. A proof of work is a partial hash collision, as introduced by Back in *hashcash* [4] and Juels in *client puzzles* [19]. The difficulty of the proof of work is dependent on the amount of the active trusts the *other* involved party has, and increased by additional trusts, until trusting an additional host is no longer economically viable. Verification of such a proof of work is essentially one hash computation, and thus a very cheap operation.

Each STUNT-capable host has to keep track about any activity regarding the trust relationships in an audit trail. The proof of work is bound to a certain state in the audit trail, such that clients can assess whether the difficulty of the proof of work was adequate, given the number of active trusts of the other party at that time.

To ensure the freshness of an active relationship, periodic updates are sent in intervals chosen by the parties themselves during their risk assessment. Clients will only consider trust relationships as valid, when both the proof of work verifies correctly and the relationship is proven to be recent. Therefore, revoking an existing relationship means to stop sending further updates. The mutuality of trust relationships between service providers is to account for the backfiring principle. Misbehaving hosts harm themselves, because they will most likely have their trust relationships dropped.

3.1 Components

The proposed system introduces an additional server service on service providers as well as client-side code to allow for browsing the STUNT network and verifying its validity. Service providers in this context are simply servers hosting web content. The system can be operated in a standalone mode, as well as in conjunction with the existing CA infrastructure. In the standalone mode, a host's certificate is self-signed. There is no CA involved, since a host is solely identified using the trust relationships as indicators for its authenticity. Alternatively, the system can well be operated next to the common certificate authority-based way of handling host authenticity. In this case, STUNT serves as an additional, independent way of assessing a host's authenticity. If operated standalone, STUNT does not necessarily need common X.509 certificates as shared data structure. However, X.509 certificates are assumed here to maintain compatibility to existing infrastructures. Regardless of the mode of operation, the STUNT service shares the *same certificate* as the web server, which ties both instances together and avoids rogue STUNT services, since clients can compare whether both given certificates match.

The additional server component is needed to handle the network-related operations, such as establishing or removing trust relationships, delivering the current status and updating the trust commitments. It is merely a service on top of a small local database that stores the relevant information, as explained later on. On the client, a different, local database is maintained, which acts as the local trust base and thus remembers the hosts users have already labeled as trusted. An elementary prototype of the system has been implemented in standard Java, using the standalone mode of operation. The proof-of-concept implementation led to several interesting insights, which influenced the system as described here.

3.2 Audit Trails

Service providers are forced to keep track of their STUNT-related activities in an audit trail that is directly embedded into the system. This audit trail is machine-readable and allows keeping track of current and former engagements. It is merely a list of the actions *initial setup*, *trust engagement attempt*, *trust engagement success*, *trust engagement rejected* and *trust engagement revoked*, alongside the corresponding timestamp and involved host, if any.

Initial Setup. A simple entry indicating the setup of the audit trail.

Trust Relationship Attempt. An entry recording whenever a host attempts to engage in a trust relationship with another host; to be appended by both parties.

Trust Relationship Success. An entry confirming a successful trust relationship establishment.

Trust Relationship Rejected. Records that the party receiving such an engagement offer denied the request.

Trust Relationship Revoked. A message indicating that a party just nullified an existing trust.

The system-wide rule for audit trails is that for each attempted trust establishment, a subsequent reject or success message has to follow, before another attempt is possible. Further attempts, who interleave a currently processed relationship attempt will be auto-denied. We use these audit trails later on to re-count the number of active trusts for a given host at any point in time.

3.3 Keys

STUNT requires two key pairs, the SSL key pair for the SSL connection and the STUNT key pair for signing elements in the protocol introduced in Section 3.4. If operated in standalone mode, a host's SSL certificate will be self-signed, because its authenticity is solely defined by the trust relationships. Regardless of the mode of operation, both the STUNT signing key pair and the SSL key pair have to be strongly bound together. Therefore, we include the public STUNT signing key in the SSL certificate by means of a certificate extension, which allows seamless distribution of the public key and separates key usage between SSL and STUNT signatures. The STUNT key pair's sole purpose is to sign messages exchanged during the STUNT protocols and may not be used for a different purpose.

3.4 Establishing Trust Relationships

In order to establish a trust relationship between two hosts, the protocol as shown in Figure 2 has to be executed. This protocol is merely the technical resemblance of a strategically important task between the two companies. At a certain level of importance, this may well involve setting up contractual agreements, which include the fingerprints of the hosts involved as well as the chosen update intervals.

Much of the responsibility of common certificate authorities gets shifted to host operators. Instead of having a CA verify the authenticity of a requester, it is now the operator's duty to do so. The backfiring principle is acting as an incentive to support careful choices, but the user interface additionally enforces certain checks using a different channel. Before an operator can establish a trust relationship, it is necessary to specify the target host's certificate fingerprint, as well as the negotiated update interval for trust commitments. If the target host's fingerprint matches, a request is sent. The recipient's STUNT service will notify the host operator of incoming requests. The protocol may as well incorporate an auto-denial mode, with only a small window of opportunity in which the service actually prompts the administrator to make a decision, and otherwise auto-denies requests. A recipient who is notified of such a request has to respond manually by accepting or denying it. If accepted, the recipient has to enter the initiator's certificate fingerprint, as well as the update interval.

The protocol itself is not secure against Man-in-the-Middle attacks. Both parties have to ensure the other's authenticity. Establishing such a relationship is considered a seldom and business-critical event. Operators are thus encouraged to exchange the certificate fingerprints using another channel, e.g. when setting up a service-level agreement. Still, the protocol needs to be run on a SSL-backed connection, using the exchanged fingerprints to ensure authenticity.

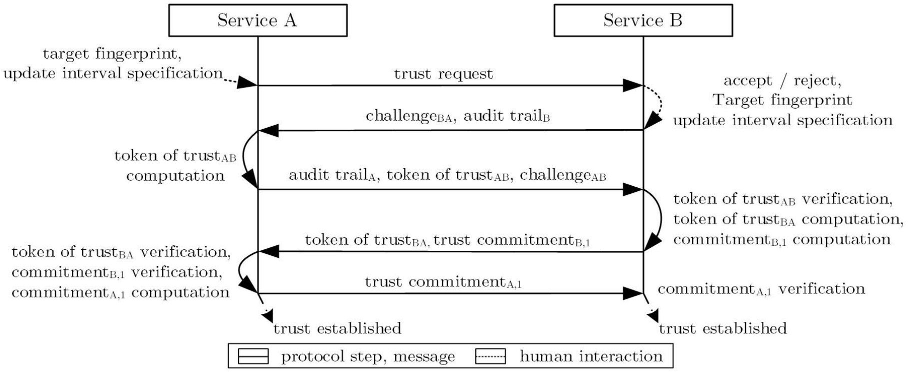
Service A
Service B
target fingerprint, update interval specification
trust request
accept / reject, Target fingerprint update interval specification
challenge$_{BA}$, audit trail$_{B}$
token of trust$_{AB}$ computation
audit trail$_{A}$, token of trust$_{AB}$, challenge$_{AB}$
token of trust$_{AB}$ verification, token of trust$_{BA}$ computation, commitment$_{B,1}$ computation
token of trust$_{BA,}$ trust commitment$_{B,1}$
token of trust$_{BA}$ verification, commitment$_{B,1}$ verification, commitment$_{A,1}$ computation
trust commitment$_{A,1}$
commitment$_{A,1}$ verification
trust established
trust established
protocol step, message
human interaction

Fig. 2. The trust engagement protocol

Suppose service B accepts the request, which is only possible if and only if the window of opportunity is open, there is no second pending request, and the host is not already trusted. In case of acceptance, a random challenge is generated to guarantee the freshness of the computed proof of work. This challenge, as well as the host's current audit trail are sent in response, if the entered certificate fingerprint matches. Service A fetches the response as well as service B's certificate and computes a *token of trust* for A and B. The unidirectional token of trust, as sent from A to B, is defined as follows:

$$\text{token of trust}_{AB} = (\left\{ \begin{array}{l} H(\text{cert}_A \mid\mid \text{cert}_B \mid\mid \text{challenge} \mid\mid \text{padding}), \\ \text{hash chain(audit trail}_B), \\ \text{challenge}, \\ \text{padding}, \\ \text{update interval} \end{array} \right\}, \sigma_A(tot_{AB}))$$

$\sigma_A(tot_{AB})$ refers to the signature over the whole structure, signed by host A using the corresponding STUNT signature key. cert$_X$ refers to the certificate of host X. H is a standardized, collision- and preimage-resistant hash function, for example SHA-256.

The proof of work is the digest generated by H. Computing a token of trust is thus a computationally expensive task of increasing difficulty. The input of H is prefixed, and a padding that leads to the desired output hash has to be found. The prefix consists of the certificates of both parties and the challenge that was randomly generated to prove freshness. The party which computes the token then has to find a padding, such that the resulting digest has n leading zeros, where n is dependent on *the other* party's number of currently active hosts. n is adjusted by a predefined rule, to ensure that the problem's difficulty is kept within computationally feasible bounds for the number of desired maximum hosts. Section 3.8 discusses this property in more detail. The more active trusts the other party has, the harder is the computation of the token, up to the point where it gets computationally infeasible to add another trust relationship. On the other hand, computing a token for a less-prominent partner is cheaper. This property leads to increased workload when one wants to bond with an already well-trusted node, but eases finding partners for new nodes, such as newly founded companies, for example.

The token is bound to a host's current state by including a hash chain over that host's current audit trail. A hash chain denotes the result of a successive computation of a hash over a certain list of elements. For the audit trail, it would mean to compute $hashchain = H(... \mid\mid H(\text{entry2} \mid\mid H(\text{entry1})))$ over all audit trail entries, including the timestamps, actions and involved hosts.

The generated token of trust consists of the proof of work, the hash chain over the audit trail, the used padding, the originally given challenge and the predefined update interval, as well as a signature over these fields. The information in the token of trust allows clients to verify the correctness of the claimed relationship. The challenge and padding allow to recompute the proof of work to determine whether the problem was adequately difficult upon the trust relationship establishment.

A token of trust seals a unidirectional trust relationship from host A to host B and is bound to the audit trail when it was created. Both sides' tokens of trust together are the expression of trust between the hosts' operators. Upon establishment, a token of trust is verified in the same way as in the client-side verification procedure described in Section 3.6.

3.5 Revocation

STUNT uses an active, whitelisting-based revocation scheme. For a trust relationship to be valid, both a valid token of trust and a fresh trust commitment is required. Trust commitments are messages sent to all directly trusted hosts in the previously negotiated update interval t. A trust commitment, as sent from A to B, looks as follows:

$$\text{trust commitment}_{AB} = \sigma_A(\sigma_A(\text{token of trust}_{AB}), \text{timestamp})$$

It is thus merely a message containing a timestamp of when the message was created, the signature of the previously sent token of trust, and a signature over the message itself. When receiving such a message, hosts store the latest commitment message from the originating host and send it to clients upon request, as discussed later on. For the trust relationship to be considered valid, the latest trust commitment needs to be no older than t.

3.6 Client-Side Verification

Client-side verification differs, depending on whether the client already considers the host in question as trusted or not. For new hosts, the verification works as shown in the pseudocode of Listing 1.

New Hosts. First of all, certain information has to be fetched from the host in question. That is, the host's certificate, its audit trail and the information about all directly trusting hosts, referred to as neighbors. The certificates from both the STUNT-service and the actual service to be used, e.g. the web server, are checked for equality beforehand, which is omitted in Listing 1. Then, the verifier has to fetch all neighbor certificates to be able to verify the token of trust signature as returned from the host in question, given that the corresponding given trust commitment is not outdated.

Algorithm 1. VERIFY_NEIGHBORS(host)

```
host_cert, audit_trail ← GET_CERTIFICATE_AUDIT_TRAIL(host)
neighbor_info ← GET_NEIGHBORS(host)
for all neighbor in neighbor_info do
    tt ← neighbor.trust_token
    neighbor_cert ← FETCH_CERTIFICATE(neighbor.IP)
    tc ← neighbor.trust_commitment
    if not RECENT(tc.timestamp, tt.interval) then
        return false // outdated commitment
    end if
    if not VERIFY_TOKEN_SIG(tt, neighbor.token_signature, neighbor_cert.kpub) then
        return false // invalid token signature
    end if
    trusts ← RECOUNT_ACTIVE_TRUSTS(audit_trail, tt.hashchain)
    if trusts == -1 then
        return false // invalid hash chain
    end if
    if not tt.proof_of_work == H(neighbor_cert, host_cert, tt.random, tt.challenge) then
        return false // invalid proof of work
    end if
    if not LEADING_ZEROS(tt.proof_of_work) == $n$ then
        return false // inadequate problem hardness
    end if
    if not tc.token_signature == neighbor.token_signature then
        return false // commitment for wrong token
    end if
    if not VERIFY_COMMITMENT(tc, neighbor_cert.kpub) then
        return false // invalid commitment signature
    end if
end for
```

Afterwards, element-wise hash computation and concatenation is performed on the given audit trail entries until the hash chain from the token of trust is met. If it is never met, either the audit trail was modified or an invalid hash chain was given. Otherwise, one is able to re-count the number of active trusts until this very state, because all successful additions as well as all revocations are recorded in the audit trail.

With that information, the verifier can recompute the proof of work using both involved certificates, as well as the given random padding and challenge. The resulting digest's leading zeroes are then compared against the number of active trusts at the time of trust establishment to verify whether an adequately hard problem was solved during the token of trust creation. Finally, the trust commitment is verified by checking the signature and comparing the token signatures.

The user is only presented with the trust relationships if these verifications succeed. Otherwise, the client might abort prematurely and issue a warning. The system's purpose is thus to ensure that the presented information was indeed created at the neighbor's will and is not entirely made up. Whether or not users decide that the presented cluster of nodes is reasonable and to be trusted is decoupled from the system and might differ from one user to another.

Verification of these yet unvisited hosts consists of many steps. Nevertheless, the individual steps are computationally cheap and the number of maximum active trusts is bound by the proof of work complexity to some intentionally low number, such that the total workload is considered negligible.

To support the claim of cheap verification, we simulated the verification procedure for newly visited hosts on a Google Nexus One mobile phone with a 1 GHz CPU and 512 MB of memory, using Android[5] 2.3 as operating system. The verification of the immediate trust network of a host having 10 trust relationships and an audit trail length of 100 entries takes not more than 132.85 ms on average, using 100 iterations. We intentionally used an outdated, constrained device, and desktop verification times are expected to be way below that value. Furthermore, we assumed a scenario with a high number of incoming trust relationships (10), while we expect the regular case to have only 5-6 incoming trust relationships to further exacerbate the complexity of the problem. Therefore, the verification time solely depends on network delays, with the actual computation time being negligible.

Subsequent Visits. Assessing whether or not a specific newly visited host will be added to one's personal trust base is considered a rare event, once the initial trust base is established and contains typical, regularly visited hosts. In contrast, ensuring that subsequent visits are indeed addressing the very same host might occur much more frequently.

Adding a host to the personal trust base means to store its certificate, as well as all given tokens of trust and commitment timestamps with the corresponding neighbors and neighbor certificates in a local database. Assuming that the user revisits a previously added host and that the last stored timestamp is outdated, then the client fetches all recent commitments from the server. For each commitment, the client checks whether it originates from a previously stored neighbor, verifies the commitment signature, checks if the timestamp is recent and whether or not the token signature matches the stored one. Thus, on subsequent visits it is not necessary to connect to any neighbors. There are basically four cases that might occur in this verification:

- The host's state is exactly as it was upon accepting it. That is, all the given information matches up with the local database. In that case, clients accept the host without any further user interaction.
- The host established one or more new trusts with other hosts. Our approach follows the logic that having more trusts expressed by others is a good thing.

[5] `http://android.com`

Therefore, the user is not disturbed. Instead, a small visual indication will be shown, with the optional possibility to re-assess the new trust network.

- The host lost one or more of the trusted hosts since it was labeled as trusted by the client. This is critical. For example, due to losing the trust relationships because of a compromization. Hence, the client should interfere and warn the user with details about the left nodes and why this might be a problem.
- One or more of the involved hosts changed their certificate. Unfortunately, this means that the stored information is invalidated and the host(s) have to be re-assessed.

The storage overhead from adding this many hosts to the personal trust base is negligible as well. We estimated the storage space required for 50 hosts, with each host having 10 distinct trust relationships and all involved hosts using RSA key lengths of 4096 bit. The proof-of-concept implementation uses a local SQLite[6] database at the client to store the trusted hosts information. We consider having 50 manually asserted, trusted hosts a high estimate for a regular user. Still, it takes no more than about 2.3 MB of storage space.

3.7 User Interface

One of the most important aspects of the system is the network explorer part of the user interface. Figure 3 shows the network explorer view from the proof-of-concept implementation, to give of an example on how such a browser might look like. The central node is the one that the user is about to visit. The surrounding nodes are hosts whose operators expressed their trust to the host in question. Upon hovering over a node, the user is able to view the website or explore the trust network deeper. After the user finished exploring the network and made up her mind, she is able to select one or multiple nodes to be labeled as trusted. The ability to add multiple hosts in one go is a convenience feature to minimize both redundant evaluations and interfering the user. Note that trust relationships are not transitive. Instead, already trusted hosts are visually indicated, such as `alice.local` in the figure.

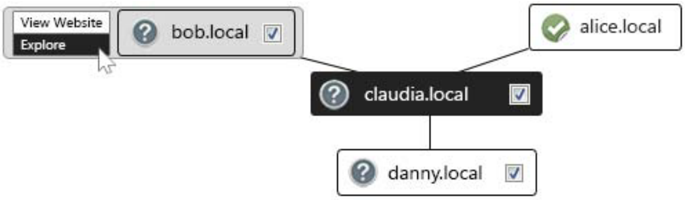

Fig. 3. STUNT network browser example

[6] `http://sqlite.org`

Whether a user interface like this will lead to the desired results, or other methods of presenting the information will be required is not yet determined and will need an in-depth usability study. A desired result would be to get users to really think about whether the presented network of trusted hosts makes sense and is reasonable, without them getting annoyed.

3.8 Limiting Trusts

The number of active trust relationships is intentionally limited to a very small number. The reason for employing a cost function instead of limiting the number of trusts to a domain-wide value, such as five hosts for example, is to avoid fast switching of trust relationships and thus underline their long-term nature.

The limitation per se serves two purposes, namely to keep such a trust relationship a scarce and thus valuable resource, and to allow users to comprehend the whole cluster of directly connected hosts on first sight. Therefore, the idea is to limit these direct connections to a value as low as about five hosts. Achieving the desired number of maximum hosts is a matter of parametrizing the scaling factor n, which defines how many leading zeros are required for the proof of work at a certain state.

The proof of work computation is flexible and thus scalable. By altering the starting offset and the rule for n, varying difficulties can be achieved. To give an example, we assume a hash function with a preimage resistance strength of $2^{\text{digest length}}$, such as the SHA family [9]. Furthermore, we assume a device capable of performing 50 million hashes per second. To delimit the maximum active trust relationships to five hosts, one example parametrization might look as depicted in Figure 4.

The rule to increase the difficulty in this example is set as follows: $n_0 = 36, n_{i+1} = n_i + i + 1$. Thus, the difficulty is increased dependent on an initial offset and the number of already-active trust relationships. The highlighted datapoints denote the required leading zeroes for five trusts. Using the assumptions above, the proof of work computation would need up to 0.4, 0.8 and 3.1 hours for the first three trust relationships, up to one day for the fourth one and up to 16 days[7] for the fifth one. Since the complexity is increased with the number of already-active trusts, a sixth trust relationship would then already require a proof of work that takes up to 521 days to compute, which is far from reasonable from an economic point of view.

Note that the number of required zeros is reduced again when revoking a trust relationship, because there is a corresponding revocation entry.

[7] These are upper bars, and one can always be lucky. However, even in the first case with $n_0 = 36$, given all possible output hashes, the probability to hit a correct hash is about $1.46 \cdot 10^{-11}$. For a 50% chance of getting a valid hash, half the time is needed.

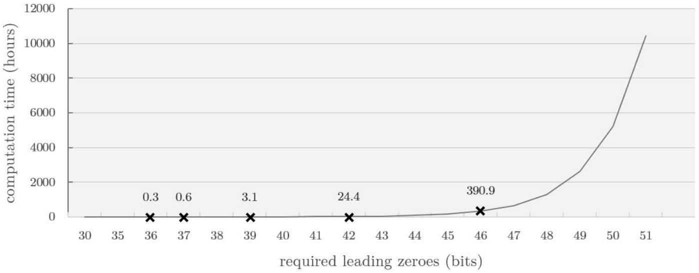

Fig. 4. Proof of work sample parameterization

3.9 Trust Communities

Trust tokens and thus the trust relationships are bound to a certain audit trail by the hash chain. While operators cannot alter an audit trail later on, nothing prevents them from starting a second one from scratch. Maintaining several audit trails lets operators increase and *categorize* their trust relationships. For example, a company might operate one category containing reference clients and another category containing suppliers, each limited to n hosts, where n is computationally bound by the proof of work computation complexity. From a user's point of view, these categories could be specially highlighted in the browser, or there might be filters in place, allowing users to explore only a certain category.

4 Security Analysis

The following informal security discussion shows attack vectors regarding STUNT and follows the adversary model by Dolev and Yao [10]. According to the model, an adversary is assumed to have full network control, but no host itself is compromised. Several possible vectors for mounting Man-in-the-Middle (MitM) attacks exist in this scenario, which are now discussed briefly. A successful MitM attack allows adversaries to read and modify exchanged messages without being detected by either the client or the server involved. Impersonating the target host without relaying the messages to the actual target is another possible approach, but treated as the same in this analysis, as there is no difference in how this attack is mounted. In this analysis, we consider only server impersonation, as STUNT is about authenticating server entities. Thus, an attacker has no gain when impersonating a client. Since full network control is assumed, an attacker can impersonate a server from both the clients' and other servers' perspective. For example, impersonating a server from a client's viewpoint is possible on public wifi hotspots. However, it is also thinkable in a larger context, that entire companies want to monitor their employees' secure communication or countries eavesdrop on citizens.

It is important to remember that both a server's STUNT- and webserver use the same certificate for authenticating the communication. Therefore, an adversary redirecting only the STUNT requests to some spoofed service will not go unnoticed, as the certificates will no longer match. Hence, adversaries are forced to spoof both services in conjunction. Furthermore, host operators are encouraged to take trust relationship decisions seriously. There is only a limited number of possible mutual trust relationships, and establishing them is computationally expensive and thus costly. Moreover, establishing a trust relationship needs an agreement using a second communication channel, for example a contract.

Within this setting, we discuss two possible attacks, which we refer to as *replacing* and *remodeling.*

Replacing. Full network control allows an attacker to impersonate a server in the network. An additional server with the same DNS name as an existing one can be introduced, and all client requests to the real host will be redirected to the impersonated one. The impersonated server can then be used to lure other operators into establishing trust relationships with it. Given that other operators agree on these trust relationships, an adversary might be able to attach a malicious node to the otherwise legitimate trust relationship network. Due to the same DNS name, the maliciously introduced server *replaces* the original one.

There is also a weaker form of *replacing*, which works having only the server's link under control, whereas the client's connection remains untouched. However, it requires attackers to choose a new, available DNS name and can thus only introduce new, seemingly trustworthy hosts.

The server operators are responsible for authenticating other servers, otherwise it is theoretically possible for an adversary to gain trust relationships. The system requires operators to take special care whom they trust. The expensive creation and mutual nature of trust relationships emphasize the importance of this decision. Whenever a host operator decides to trust another host, she will receive the trust vice versa. There is only a limited amount of incoming trust relationships, and adding an untrusted host backfires, because it reduces the likeliness of being accredited as trustworthy by users. Since adding trusts is expected to be a rare event, operators are asked to exchange the expected fingerprints on a second channel, which counters rogue trust relationships. If the attacker manages to set up such a seemingly trustworthy host with the same DNS name, there would still be a warning if the attacked user has visited the real host before. A warning is issued, because the presented and the previously stored certificate from the original server would not match any more. Furthermore, such trust relationships are not expected to last for long, since the fraud is likely to be detected soon.

Remodeling. Instead of getting other operators to establish a desired trust relationship, an attacker might as well fake a network of a trusted server by adding a number of self-established hosts. We call this *remodeling*, since the attacker tries to copy an existing network of trust, or to create a similar one. This attack requires only client-side networking control, but is a lengthy and expensive task due to the proofs of work. Furthermore, its success is highly dependent on the user's reaction, because it can not be known how much effort the user puts into investigating the presented network. A *remodeled* trust network is assumed to be very small, because each additional host to fake is computationally expensive for the adversary. Since the user can browse the network infinitely deep, the deception will eventually come to light. In the large-scale case of companies and countries, such fake networks get more realistic. However, their use is limited, which renders the expense/gain ratio very low. They will not last for long, because people typically visit hosts like their online banking not just once and from different places. Their clients store the corresponding certificates and will thus issue warnings as soon as a subsequent visit yields different fingerprints. More people being affected means more gain for the attacker, but the faster it will be detected.

Furthermore, to render this vector almost completely unusable, the system can be used in conjunction with Perspectives [28], having notaries on various places in the world monitor certificate fingerprints. An additional variant would be to provide a tool which allows easy comparison of the presented trust network on a desktop client to the one that is presented on the mobile device, which usually uses a connection provided by an independent carrier.

5 Conclusion and Future Work

We propose and implement STUNT, a system that uses the concept of checking references to determine a host's trustworthiness in large-scale networks. STUNT helps the user by modeling the concept of asking for references before interacting with an unknown entity. While STUNT cryptographically ensures that no false claims are made to fake references, ultimately the user decides whether to trust a certain host. Given an intuitive and clear user interface, we believe that STUNT supports accurate trust assertions.

STUNT encourages establishing meaningful trust relationships, because trust relationships are an inherently limited, and expensive resource. Existing trust relationships can be revoked in a timely manner, due to an active, risk-, and whitelisting-based revocation scheme. Currently, no usability study has been performed, but such a study on the usability and scalability of the proposed system in large-scale environments would be an essential contribution. While STUNT is still in its infancy, we consider STUNT a solid base for future, user-centered solutions. We are confident STUNT will enable users to meaningfully question a host's authenticity, and it helps them maintaining an individual and dynamic trust base.

References

1. Abdul-Rahman, A., Hailes, S.: A Distributed Trust Model. In: Proceedings of the 1997 Workshop on New Security Paradigms, pp. 48–60. ACM Press, New York (1997)

2. Arnbak, A.M., van Eijk, N.A.N.M.: Certificate Authority Collapse - Regulating Systemic Vulnerabilities in the HTTPS Value Chain. In: Telecommunications Policy Research Conference (2012), http://www.ssrn.com/link/2012-TPRC.html
3. Artz, D., Gil, Y.: A Survey of Trust in Computer Science and the Semantic Web. Web Sem. 5(2), 58–71 (2007)
4. Back, A.: Hashcash - A Denial of Service Counter-Measure. Technical Report (2002)
5. Bartsch, S., Volkamer, M., Theuerling, H., Karayumak, F.: Contextualized Web Warnings, and How They Cause Distrust. In: Huth, M., Asokan, N., Čapkun, S., Flechais, I., Coles-Kemp, L. (eds.) TRUST 2013. LNCS, vol. 7904, pp. 205–222. Springer, Heidelberg (2013)
6. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: IEEE Symposium on Security and Privacy, pp. 164–173. IEEE Press, Washington DC (1996)
7. Blaze, M., Feigenbaum, J., Keromytis, A.D.: KeyNote: Trust Management for Public-Key Infrastructures. In: 6th International Workshop on Security Protocols, pp. 59–63. Springer, London (1999)
8. Chu, Y.-H., Feigenbaum, J., LaMacchia, B., Resnick, P., Strauss, M.: REFEREE: Trust Management for Web Applications. World Wide Web 2(3), 127–139 (1997)
9. Dang, Q., Blank, R.M., Gallagher, P.: Recommendation for Applications Using Approved Hash Algorithms. NIST Special Publication 800-107, Rev. 1. National Institute of Standards and Technology (2012)
10. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. on Inf. Theory 29(2), 198–207 (1983)
11. Ellison, C., Schneier, B.: Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. Comp. Sec. 16(1), 1–7 (2000)
12. European Network and Information Security Agency (ENISA): Operation Black Tulip: Certificate Authorities lose authority (2011), http://www.enisa.europa.eu/media/news-items/operation-black-tulip/view
13. Evans, C., Palmer, C., Sleevi, R.: Public Key Pinning Extension for HTTP, IETF Experimental Draft (2012), https://tools.ietf.org/html/draft-ietf-websec-key-pinning-04
14. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V.: The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. In: Computer and Communications Security, pp. 38–49. ACM Press, New York (2012)
15. Gutmann, P.: PKI: It's Not Dead, Just Resting. Computer 35(8), 41–49 (2002)
16. Hoogstraaten, H., Prins, R., Niggebrugge, D., Heppener, D., Groenewegen, F., Wettinck, J., Strooy, K., Arends, P., Pols, P., Kouprie, R., Moorress, S., van Pelt, X., Zheng, H.Y.: Black Tulip - Fox-IT Report of the Investigation into the DigiNotar Certificate Authority Breach (2012), http://tinyurl.com/942cr6h
17. ITU-T, Series X: Data Networks, Open System Communication and Security: Public-Key and attribute certificate frameworks. ITU-T Recommendation X.509 (2009)
18. Jøsang, A., Ismail, R., Boyd, D.: A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems 43(2), 618–644 (2007)
19. Juels, A., Brainard, B.: Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In: Kent, S. (ed.) Network and Distributed System Security Symposium, pp. 151–165 (1999)

20. Laurie, B., Langley, A., Kasper, E.: Certificate Transparency, IETF Experimental Draft (2012), `https://tools.ietf.org/html/draft-laurie-pki-sunlight-05`
21. Marlinspike, M., Perrin, T.: Trust Assertions for Certificate Keys, IETF Experimental Draft (2013), `https://tools.ietf.org/html/draft-perrin-tls-tack-02`
22. Microsoft Corporation: Windows and Windows Phone 8 SSL Root Certificate Program, Member CAs (2013), `http://tinyurl.com/c2qa5nh`
23. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008), `http://bitcoin.org/bitcoin.pdf`
24. Soghoian, C., Stamm, S.: Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 250–259. Springer, Heidelberg (2012)
25. Sotirakopoulos, A., Hawkey, K., Beznosov, K.: On the Challenges in Usable Security Lab Studies: Lessons Learned from Replicating a Study on SSL Warnings. In: 7th Symposium on Usable Privacy and Security, pp. 1–18. ACM, New York (2011)
26. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In: 18th USENIX Security Symposium, pp. 399–416. USENIX Association, Berkeley (2009)
27. Thoughtcrime Labs: Convergence - an Agile, Distributed and Secure Strategy for Replacing Certificate Authorities (2013), `http://www.convergence.io`
28. Wendlandt, D., Andersen, D., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In: USENIX Annual Technical Conference, pp. 321–334. USENIX Association, Berkeley (2008)

What Public Keys Can Do for 3-Party, Password-Authenticated Key Exchange

Jean Lancrenon

Université du Luxembourg,
Interdisciplinary Centre for Security, Reliability and Trust,
6 rue Richard Coudenhove-Kalergi, L-1359 Luxembourg,
Luxembourg
jean.lancrenon@uni.lu

Abstract. We study three-party, password-authenticated key exchange protocols where the trusted third party has a high-entropy private key to which corresponds a public key. In this scenario we can maintain the user-friendliness of password authentication while provably achieving security properties that ordinary password-authenticated key exchange protocols cannot, namely resistance against key compromise impersonation and a special form of internal state revealing. We define security models tailored to our case and illustrate our work with several protocols.

Keywords: Password-authenticated key exchange, public-key cryptography, provable security, simulation-based security, internal state.

1 Introduction

1.1 Password-Authenticated Key Exchange

2-PAKEs and 3-PAKEs. This article's main focus is on the design and analysis of Password-Authenticated Key Exchange (PAKE) protocols. The goal of a PAKE is for two users to perform a cryptographic Key Exchange (KE) that is authenticated using each user's knowledge of a *password*. In the two-party case (2-PAKE), both users share the same password, and in the three-party case (3-PAKE), each user shares its own password with a server (a trusted third party) which aids in the exchange. We will be studying 3-PAKEs in which the server also holds strong secret keying information to which corresponds a public key. We call such protocols *3-PAKEs with server private keys*, or 3-PAKE[spk]s.

Motivation and Challenges. The study of PAKEs is important because of the ubiquity of passwords in everyday applications. In contrast to cryptographically strong keys (e.g, symmetric and private or asymmetric and managed in a public-key infrastructure), passwords are easy to handle thanks to their most prominent property: they are typically *low-entropy*. However, this also makes brute force password guessing feasible. Thus, PAKEs must be designed to resist *dictionary attacks*: a well-constructed PAKE should allow at most one password to be

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 83–101, 2014.

tested per online attempt (*online* dictionary attack resistance), and its message exchange should not leak *any* information on the password (*Offline* Dictionary Attack - or OffDA - resistance).

Our Contributions, in a Nutshell. Most of the research done in the last decade has provided provably secure 2-PAKEs; 3-PAKEs have been much less studied. Most notably, there seems to be no *proven secure* 3-PAKE[spk]. We propose to fill this gap by describing several security models and protocols proven secure in these models.

Our work shows that adding a public/private key pair first makes protocol design much simpler, as it becomes possible to rely on simple, well-understood cryptographic notions. Secondly, we exhibit protocols that enjoy desirable KE properties that are *unobtainable in PAKEs that only use passwords.* These properties are resistance against key compromise impersonation and resistance against (a special form of) internal state revealing. Third, none of these sacrifice usability.

1.2 Related Work

2-PAKEs. Bellovin and Merrit [6] were the first to consider dictionary attacks in 2-PAKEs, and the first to propose a solution, followed by Jablon [20]. Lucks [24] gave formal definitions of OffDA resistance. Halevi and Krawczyk [18] and Boyarsky [7] provided definitions and protocols for 2-PAKEs where one of the parties is a server with strong keying information. Many other protocols have been proposed, e.g. [1,4,8,10,11,22,13,14,17,19,23,21], most of them provably secure. Bellare et al. [4] adapted to 2-PAKEs the now widespread indistinguishability-based security model of [5]. Boyko et al. [8] did this with the simulation-based models in [28] and [3]. In [1,11,22,13,14,17,23,21] are practical 2-PAKEs secure without random oracles using Cramer and Shoup's smooth projective hashing [12], and assuming a Common Reference String (CRS). There are also theoretical results [15,16] showing that efficient but impractical secure 2-PAKEs can be constructed with neither random oracles nor CRSs. Designing a practical 2-PAKE secure in the standard model with no CRS is an open problem.

3-PAKEs. The first to consider 3-PAKEs seem to be Steiner et al. [29]. They require that the server only know the passwords. Similar protocols can be found in [2,27,9,30,31]. In [2] the first security model for 3-PAKEs is defined following the approach of [5,4], the most commonly used security model for 2-PAKEs is strengthened, and a method to get 3-PAKEs from 2-PAKEs is proposed. Besides [2] and [31], none of the cited works have formal security definitions. Other solutions require the trusted server to hold a secret key of its own, e.g. [26,32]. Neither of these protocols have formal security proofs.

1.3 Organization of the Paper

In section 2 we explain why we think 3-PAKE[spk]s are worth considering. Next, section 3 states the security properties we can expect of a 3-PAKE[spk] and gives an overview of the ideal-world simulation paradigm. Sections 4, 5, and 6 then respectively exhibit the static, password-adaptive, and password-and-state-adaptive network adversary models, the latter two being each punctuated with a protocol and some comments. Finally, section 7 concludes the paper. Proof ideas and computational assumptions are in the appendix; the detailed proofs, which are very long, are in the full paper.

2 Why 3-PAKEs with Server Private Keys?

2.1 Retaining User-Friendliness

In a 3-PAKE[spk], in addition to the users' passwords, the trusted server $\mathcal{T}$ holds strong, secret keying material $sk_{\mathcal{T}}$ to which corresponds a public key $pk_{\mathcal{T}}$ available to each user. It is legitimate to wonder why a string like $pk_{\mathcal{T}}$ is making an appearance at the users' end: after all, is not the whole point of using passwords to get rid of such cumbersome data? The answer is that users will not have to know it because since it is the same for every user, it can be hardwired into the protocol specification. The application thus retains its user-friendliness.

This concept is not new in PAKE research. All *practical* 2-PAKEs that are proven secure *without random oracles* use this hardwiring technique because they rely on a *Common Reference String* (CRS) known to all users. A CRS is basically a long public string that is generated in some secret manner. In PAKEs, they tend to appear as public keys to public-key encryption schemes; the "secret" part is the decryption key which must be immediately destroyed, for any entity that gets it can undermine the whole system undetected.

2.2 Proving Security without Idealized Assumptions

Aside from the work of Abdalla et al. [2], all known 3-PAKEs either lack a security proof, or rely on idealized assumptions. In [2] the authors devise a way to generically construct a 3-PAKE from a 2-PAKE and prove its security in the sense that if the 2-PAKE is standard-model-secure, then so is the 3-PAKE. We claim that given the current state-of-the-art in 2-PAKE research, our method for obtaining 3-PAKEs secure in the standard model is a good, and possibly more efficient, alternative. We reason as follows.

We already mentioned that all practical standard-model-secure 2-PAKEs use a CRS. Some entity has to generate this CRS and be trusted to destroy the corresponding secret. If we construct a 3-PAKE generically from a CRS-based 2-PAKE, the best-placed entity for this is the server, since it is already trusted with all of the passwords. But then this server is also a very natural candidate to trust with correctly *using* strong secrets, rather than discarding them. It is, after all, a pity to not use the decryption key of a public-key encryption scheme. Thus, it makes sense to return to considering 3-PAKE[spk]s.

The upshot is that if our *starting point* is having strong server secrets, protocol design complexity drops significantly. We emphasize however that we are **not** stating that in general protocol design, a CRS-based construction can be replaced by a server with secret keys. Our reasoning is specific to 3-PAKEs.

2.3 Simpler Practical Protocol Designs

The reason for using a public encryption key as the CRS in 2-PAKEs is that no information on a password can be plucked from a *semantically secure encryption* of it, thus avoiding OffDAs. The challenge then becomes finding a way for a legitimate partner to exploit the ciphertext *without decrypting*. Subtle tools, e.g. *smooth projective hashing* [12], can be used for this, but yield complex designs.

In contrast, we let the server use the secret key, so we can rely in a straightforward way on well-known, classic primitives. The protocols we propose have users encrypt their passwords which can then be verified through decryption by the server, and the server digitally signs messages for users, who can verify the signatures. The only properties required of the primitives involve their level of security, and nothing else.

Our protocols require at most six sent messages, including key confirmation. Factoring in the additional security properties we can achieve, this is certainly not excessive: four of the messages are used to authenticate users to the server and vice-versa. The precise role of the other messages depends on the protocol. We can compare this to an instantiation of Abdalla et al.'s scheme [2] with a two-message 2-PAKE, and no key confirmation between *any* parties: this yields seven messages sent.

2.4 Capturing More Security Properties

Finally, a good reason to consider 3-PAKE[spk]s is we can prevent more attacks.

Key Compromise Impersonation. In KE, *Key Compromise Impersonation* (KCI) is said to occur if an attacker that gets a user's long-term secret can impersonate some other user to that user. 2-PAKEs cannot satisfy this property: if an attacker has a user's password, that attacker can always impersonate the *other holder* of that password to that user. At least it cannot impersonate other users to that user, as this requires compromising other passwords.

In a 3-PAKE, a user shares its password only with the server. If this password is the only data that proves to a user that it is indeed speaking to the server, then compromise of this password will allow the adversary to impersonate to that user *the server and any other user in the network at any time*, which is the worst KCI scenario possible.

We can thwart this however if the server authenticates itself to users via strong secrets. This is interesting because it is reasonable to assume that any secret at the server will be better protected than any password ever is (or will be) by a user. Our protocols either heavily restrict (5.3) or eliminate (6.3) KCI.

Unerased State Revealing. KE research nowadays often considers security notions based on revealing forms of internal state. The main idea is to let the adversary obtain certain intermediate values during a protocol run, e.g. the *ephemeral randomness* of [25], or the *unerased internal state* of [28].

This is rarely seen in PAKE research because any form of internal state revealing yields trivial OffDAs. Thus, while some works do consider revealing state (e.g. [1,10]), it includes password revealing by default.

We show that a definition of resistance against *Unerased Internal State* (UIS) revealing that does not disclose the password can be considered for 3-PAKE[spk]s by exhibiting protocols for which this query does not lead to OffDAs. Our protocols both do this, although only one fulfills the security goal (6.3).

3 3-PAKE[spk]s and Ideal-World Simulation

In the next sections we adapt the simulation-style models of Shoup [28] and Boyko et al. [8] to 3-PAKE[spk]s. We use their notations.

3.1 General System Description

The Users. The *users*, indexed by positive integers i, have identities ID_i of some fixed length. Each user i selects a private password pw_i uniformly at random from a *dictionary* D. D is a (possibly small) public set of strings of some fixed length.

The Server. *Server* $\mathcal{T}$ runs a *server key generation algorithm* $\mathcal{K}_\mathcal{T}$ once on input security parameter 1^η at the time the system is initialized to produce a pair $(pk_\mathcal{T}, sk_\mathcal{T})$. $\mathcal{T}$ keeps $sk_\mathcal{T}$ secret and $pk_\mathcal{T}$ is copied into the users' specification. The list $\{ID_i, pw_i\}_i$ is given to $\mathcal{T}$, who also keeps the passwords secret.

Statement of Goals and Desirable Properties

Protocol goal At the end of a protocol run with i, i', and $\mathcal{T}$, either i and i' used as input their passwords pw_i and $pw_{i'}$ and key $pk_\mathcal{T}$ to authenticate themselves to $\mathcal{T}$, and $\mathcal{T}$ used as input its secret $sk_\mathcal{T}$ to authenticate itself to both users, in which case i and i' should have computed a secret, random-looking, shared session key SK, or the protocol is aborted by one of the parties.

Desired security properties In [2], dictionary attacks proper to 3-PAKEs are identified: *insider attacks* in which registered users try to determine other users' passwords through protocol runs. We also take these into account.

Another security feature isolated in [2] is that of *session key privacy with respect to the server.* We assume that $\mathcal{T}$ is at worst *Honest-but-Curious* (HbC), i.e. that the only bad behavior $\mathcal{T}$ might have is eavesdropping. It is otherwise trusted to carry out the protocol faithfully. In order to minimize the trust placed in $\mathcal{T}$, we require that $\mathcal{T}$ not be able to compute the session key established during a protocol run. This separates three-party key *exchange* and *distribution.*

We will be considering *network adversaries*, which completely control all communications between parties. HbC servers are treated in the full paper. As in [2], these must be modeled separately: merging them does not make sense, for this would amount to giving the network adversary all of the secret keys.

3.2 Ideal-World Simulation Methodology

We define security via the *ideal-world simulation paradigm.* Two computational worlds are described. In the *ideal world* the protocol's goals are ideally achieved between all users in the presence of an adversary. Bad events here represent inevitable attacks on the service we are providing. In the *real world*, the protocol is executed between users in the presence of an adversary who may disturb it however he wishes according to the powers he is afforded. We say that security is achieved if for any real-world adversary running against the protocol we can construct an ideal-world adversary that behaves the same way.

Adversaries and Ring Masters. The *adversary* ($\mathcal{M}^*$ in the ideal world, $\mathcal{M}$ in the real world) plays against the *ring master* ($\mathcal{RM}^*$ in the ideal world, $\mathcal{RM}$ in the real world) whose task is to generate all of the necessary random values, and to process *operations* the adversary asks to have performed. All adversaries are assumed to be probabilistic and polynomial-time in security parameter $\eta \in \mathbb{N}$.

Transcripts. As the interaction progresses, a *transcript* ($\mathcal{IW}(\mathcal{M}^*)$ in the ideal world, $\mathcal{RW}(\mathcal{M})$ in the real world) logging the adversary's actions is built.

Definition of Security. We are now ready to define security:

We say that a 3-PAKE[spk] is secure against network adversaries if for every real-world network adversary $\mathcal{M}$ there exists an ideal-world network adversary $\mathcal{M}^*$ such that $\mathcal{IW}(\mathcal{M})$ and $\mathcal{RW}(\mathcal{M}^*)$ are computationally indistinguishable.

The adversary is in both worlds allowed to have established session keys placed in the transcript. In the real world, these are the real keys. In the ideal world they will be random, independent strings. The security definition thus captures the idea that correctly exchanged session keys cannot be efficiently told apart from random strings. This is why when following the ideal-world simulation paradigm the test query found in [5]-type models is unnecessary.

4 Static Network Adversaries

We first describe a model that captures *statically corrupted users*, i.e. parties that are registered by the adversary with passwords of its choice. Our models' main description is in this section; sections 5 and 6 just add operations. We state what operations the adversary can ask of the ring master, explain what their effects are, and with which string they are logged in the transcript. Operation names are in bold; paragraphs in italics contain additional notes.

4.1 The Static Ideal World

Static network adversary $\mathcal{M}^*$ may ask ring master $\mathcal{RM}^*$ to do the following.

Initialize Server. $\mathcal{M}^*$ starts the game by formally invoking the server controlled by $\mathcal{RM}^*$. It carries the index 0.

String logged in the transcript: $(\text{"}\textit{initialize server}\text{"}, 0)$

Initialize User, i, ID_i. $\mathcal{M}^*$ chooses an *identity* bitstring ID_i with which to initialize i. ID_i should not have been used before for another user, or for a "set password" operation (see below). Also, a *password* pw_i is chosen uniformly at random from D and assigned to i outside of the adversary's view.

Transcript: $(\text{"}\textit{initialize user}\text{"}, i, ID_i)$

$\mathcal{M}^*$ may invoke new honest players with their own passwords.

Set Password, ID, pw. $\mathcal{M}^*$ specifies an identity ID that has not previously been assigned to a user and associates to this string a password pw of its choice.

Transcript: $(\text{"}\textit{set password}\text{"}, ID, pw)$

This query allows $\mathcal{M}^*$ to put into play registered users it controls.

Initialize User Instance, (i, j), $role_{ij}$, PID_{ij}. $\mathcal{M}^*$ asks to have user i initialize an *instance* j of that user. $\mathcal{M}^*$ gives (i, j) a *role* $role_{ij} \in \{open, connect\}$, and a *partner identity* PID_{ij}. If PID_{ij} was not given to an honest user, we require that a "set password" operation on PID_{ij} has already been performed.

Transcript: $\big(\text{"}\textit{initialize user instance}\text{"}, (i, j), role_{ij}, PID_{ij}\big)$

This operation allows $\mathcal{M}^*$ to activate many different KEs for a given user. During such an exchange, a user may either be waiting for its partner to connect to it, or may be expected to connect to its partner.

Initialize Server Instance, $(0, k)$, $PIDS_{0k}$. $\mathcal{M}^*$ asks to have an instance k of the server initialized, on input a pair of *partner identities* $PIDS_{0k} = (OID_{0k}, CID_{0k})$. OID_{0k} is the *opening partner identity* and CID_{0k} is the *connecting partner identity.* These should be distinct, at least one of them is assigned to a user, and both must have passwords. Note that the server assigns to OID_{0k} the role *open* and to CID_{0k} the role *connect.*

Transcript: $\big(\text{"}\textit{initialize server instance}\text{"}, (0, k), PIDS_{0k}\big)$

This allows $\mathcal{M}^*$ to invoke server instances. Naturally, this server expects to be relaying messages between two specific users.

Terminate User Instance, (i, j)

Transcript: $\big(\text{"}\textit{terminate user instance}\text{"}, (i, j)\big)$

Terminate Server Instance, $(0, k)$

Transcript: $\big(\text{"}\textit{terminate server instance}\text{"}, (0, k)\big)$

Test Instance Password, i, $(0, k)$, pw. $\mathcal{M}^*$ inputs a user i, a not yet terminated server instance $(0, k)$ with $ID_i \in \{OID_{0k}, CID_{0k}\}$, and a password pw. It receives in return whether or not $pw = pw_i$. This is allowed only if $(0, k)$ has not completed an exchange (see below). We also make the restriction that this operation can only be done **once on input i, $(0, k)$, pw.** We will say that a password was *successfully guessed* if it has been the target of a successful "test instance password" operation. If the operation says the guess is incorrect, we will say that it *failed.*

This operation leaves no record in the transcript.

This lets $\mathcal{M}^*$ test password guesses at the server. No such tests target users because we explicitly forbid impersonating the server; this reflects it holding a private key that network adversaries do not have access to. This formally excludes protocols that use the passwords to authenticate the server to users.

Notice that if $\mathcal{M}^*$ tries a guess on one end of the server instance, ***it still has the possibility to attempt a guess at the other end of this same***

instance. *Thus, there are scenarios where $\mathcal{M}^*$ could run* ***two*** *"test instance password" operations on* ***one*** *initialized instance. This cannot happen in [8].*

Exchange Completed, $(0,k)$**.** $\mathcal{M}^*$ specifies a not yet terminated server instance $(0,k)$ and indicates that it has completed its role in the exchange. This requires that no **failed** "test instance password" operation was conducted on $(0,k)$ targeting a component of $PIDS_{0k}$ that is assigned to an initialized user.

Transcript: $\big($"*exchange completed*"$, (0,k)\big)$

$\mathcal{M}^*$ *may stipulate when a server instance has served its purpose in an exchange. Accordingly, it can only occur if both password checks have passed.*

Start Session, (i,j)**.** $\mathcal{M}^*$ specifies a not yet terminated user instance (i,j) and asks that a session key be given to it. (i,j) gets a *connection assignment*:

- **Open for Connection from** (i',j') **through** $(0,k)$**.** For this we require $PID_{ij} = ID_{i'}$ for some initialized user i', $role_{ij} = open$, (i',j') to have been initialized, $PID_{i'j'} = ID_i$, $role_{i'j'} = connect$, $(0,k)$ to have been initialized, and $PIDS_{0k} = (ID_i, ID_{i'})$. $\mathcal{RM}^*$ selects a session key SK^*_{ij} uniformly at random; we now say that (i,j) *is open for connection from* (i',j') *through* $(0,k)$.

There are not too many prerequisites for an instance to be open for connection from another instance, because we authorize implicit authentication. In this case, the user that is the last to send a message can very well have only received replayed material. The resulting session key should still be uncomputable by the adversary. The server instance is required to mirror the fact that before computing a session key, the opening instance must be sure that its partner has been authenticated, which can only be done through the server.

- **Connect to** (i',j') **through** $(0,k)$**.** This requires $PID_{ij} = ID_{i'}$ for some initialized user i', $role_{ij} = connect$, (i',j') to have been opened for connection from (i,j) through $(0,k)$, and $(0,k)$'s exchange to have been completed after (i',j') has been initialized. $\mathcal{RM}^*$ sets $SK^*_{ij} \leftarrow SK^*_{i'j'}$.

For one instance to connect to another, there are more requirements. Both users must have had their passwords checked, so the server's exchange must have been completed. Also, the connecting instance must have been initialized before the partner was opened for connection and the server has completed its exchange.

- **Exposed through** $(0,k)$**.** If $role_{ij} = open$ (resp., $connect$), this is allowed if $(0,k)$ has $PIDS_{0k} = (ID_i, PID_{ij})$ (resp., $(0,k)$'s exchange has been completed with $PIDS_{0k} = (PID_{ij}, ID_i)$), and either PID_{ij} is not the identity of a user, or PID_{ij} is the identity of a user whose password was successfully guessed, or pw_i was successfully guessed. In this case, $\mathcal{M}^*$ specifies SK^*_{ij}.

This lists the conditions that must be met for an instance to be sharing a key with $\mathcal{M}^*$. Either its partner is a statically corrupted user, or it is a user whose password was guessed, or its own password was guessed. The connection still needs to have gone through a server instance with appropriate partner identities, since $\mathcal{M}^*$ cannot impersonate the server. Note that these rules do not forbid *opening or connecting users even if a password has been guessed; an honest exchange can occur even if the adversary knows a password.*

Conservative vs. liberal exposure: [28] describes two different exposure rules. The **liberal** rule is the one above while the **conservative** rule stipulates that exposing (i, j) **is not allowed if only** pw_i **was guessed.** Security under one or the other provides different guarantees. This distinction is absent from [8] because it only makes sense when the communicating parties have **different long-term keys**, which is not the case for 2-PAKEs.

We finally require that connection assignments be efficiently computable from the transcript up to the current "start session" operation. This reflects the fact that a rule is in place to determine how conversations match (how this is done is not relevant to the ideal service, but is crucial for concrete protocols) and forces uniqueness of connection assignments.

Transcript: $\big("start\ \ session", (i, j)\big)$

Reveal Session Key, (i, j)**.** $\mathcal{M}^*$ specifies an unexposed user instance (i, j) that holds a session key, and receives SK^*_{ij}.

Transcript: $\big("reveal\ \ session\ \ key", (i, j), SK^*_{ij}\big)$

This models session key leakage; it should not affect the security of new keys.

Impl, *string***.** $\mathcal{M}^*$ simply adds a string to its transcript. This is to make sure that the transcripts cannot be distinguished because of a difference in syntax. Also, these operations provide information to compute connection assignments.

Transcript: $("impl", string)$

This completes our description of the ideal world adversary's actions.

4.2 Some Further Explanations

Dictionary Attacks. Testing a password can only be done once per password held by a server instance, and is the only way to verify guesses, so online trials are indeed limited to one per one of the two passwords at a given instance. Since testing is not allowed after a complete exchange, OffDAs are also avoided. Finally, the server can be between an honest user and a corrupted one, covering insiders.

Key Compromise Impersonation. Observe the difference between the liberal and conservative rules: the first allows KCI, the second does not. **Prot1** (5.3) is secure under the liberal rule, and **Prot2** (6.3), under the conservative one.

Remarks. Despite authorizing implicit authentication, we do not need the "dangling" assignments of [8]. These are used because mutual authentication for 2-PAKEs is equivalent to key confirmation since there is no way for one party to verify the other's password message by message. 3-PAKE[spk]s do not suffer from this: our protocols do actually perform message-by-message authentication.

As in [8], passwords are incorporated in the ideal world. Otherwise, we would have to account for the adversary's non-negligible advantage in guessing passwords online: security would be defined by asking that $\mathcal{IW}(\mathcal{M}^*)$ and $\mathcal{RW}(\mathcal{M})$ be "only negligibly more distinguishable than $\frac{N_o}{\#D}$", where N_o is the number of real-world online guesses. This is arguably more elegant because in the ideal service of *any* KE, the authentication process is irrelevant: entities that authenticate correctly by whatever means end up sharing a key, and otherwise do not.

4.3 The Static Real World

Now we describe a real static network adversary $\mathcal{M}$'s queries.

Initialize Server. $\mathcal{M}$ starts the game by invoking $\mathcal{T}$, identified with index 0. $(pk_{\mathcal{T}}, sk_{\mathcal{T}})$ is generated by $\mathcal{RM}$, $pk_{\mathcal{T}}$ is given to $\mathcal{M}$, and $\mathcal{RM}$ will run $\mathcal{T}$.
Transcript: $(\text{"}\textit{initialize server}\text{"}, 0)$ and $(\text{"}\textit{impl}\text{"}, \text{"}\textit{server public key}\text{"}, pk_{\mathcal{T}})$

Initialize User, i**,** ID_i**.** $\mathcal{M}$ chooses an identity ID_i for i. ID_i should not have been used before for another user or a "set password" operation. A password pw_i is chosen uniformly at random from D outside of $\mathcal{M}$'s view.
Transcript: $(\text{"}\textit{initialize user}\text{"}, i, ID_i)$

Set Password, ID**,** pw**.** $\mathcal{M}$ specifies an identity ID that has not already been attributed to a user, and a password pw.
Transcript: $(\text{"}\textit{set password}\text{"}, ID, pw)$

Initialize User Instance, (i,j)**,** $role_{ij}$**,** PID_{ij}**.** $\mathcal{M}$ initializes (i,j), with $role_{ij} \in \{open, connect\}$ and a partner PID_{ij}, which must have a password.
Transcript: $\big(\text{"}\textit{initialize user instance}\text{"}, (i,j), role_{ij}, PID_{ij}\big)$

Initialize Server Instance, $(0,k)$**,** $PIDS_{0k}$**.** $\mathcal{M}$ initializes $(0,k)$ on input $PIDS_{0k} = (OID_{0k}, CID_{0k})$. OID_{0k} is the opening partner and $CID_{0k} \neq OID_{0k}$ is the connecting one. One at least is a user, and both have passwords.
Transcript: $\big(\text{"}\textit{initialize server instance}\text{"}, (0,k), PIDS_{0k}\big)$

Deliver User Message, (i,j)**,** $InMsg$**.** (i,j) must be initialized. $\mathcal{M}$ specifies an *incoming message* $InMsg$ which (i,j) processes according to the protocol. (i,j) produces an *outgoing message* $OutMsg$ and reports its *status* $status_{ij} \in \{accept, continue, reject\}$ to $\mathcal{M}$. If (i,j) accepts, it generates a session key SK_{ij} and halts. If it continues, it has not generated a key yet and is expecting another message. If it rejects, it terminates without generating a key.
Transcript: $\big(\text{"}\textit{impl}\text{"}, \text{"}\textit{message}\text{"}, (i,j), InMsg, OutMsg, status_{ij}\big)$
If the instance accepts, add $\big(\text{"}\textit{start session}\text{"}, (i,j)\big)$.
If the instance rejects, add $\big(\text{"}\textit{terminate user instance}\text{"}, (i,j)\big)$.
This lets $\mathcal{M}$ *actively interfere in protocol runs by, e.g. interleaving them, injecting messages of its own, or forwarding messages as intended.*

Deliver Server Message $(0,k)$**,** $InMsg$**.** $(0,k)$ must be initialized. $\mathcal{M}$ specifies an incoming $InMsg$ which is treated according to the protocol. The $status_{0k} \in \{accept, continue, reject\}$ is reported, and an outgoing $OutMsg$ is produced. If $(0,k)$ accepts, it has delivered all of the messages it expects to and halts. If it continues, it is waiting for another message. If it rejects, it terminates.
Transcript: $\big(\text{"}\textit{impl}\text{"}, \text{"}\textit{message}\text{"}, (0,k), InMsg, OutMsg, status_{0k}\big)$

If the instance accepts, add $\big(\text{"}exchange\ \ completed\text{"}, (0,k)\big)$.
If the instance rejects, add $\big(\text{"}terminate\ \ server\ \ instance\text{"}, (0,k)\big)$.

Reveal Session Key, (i,j)**.** $\mathcal{M}$ specifies a previously initialized user instance (i,j) that has accepted. It receives the session key SK_{ij}.
Transcript: $\big(\text{"}reveal\ \ session\ \ key\text{"}, (i,j), SK_{ij}\big)$

Adversary Coins. When $\mathcal{M}$ halts, the last transcript log is
Transcript: $(\text{"}impl\text{"}, \text{"}adversary\ \ coins\text{"}, coins)$
where *coins* holds all of $\mathcal{M}$'s random choices during the interaction.

5 Password-Adaptive Network Adversaries

We now add a *"reveal password" operation* to model password leakage outside of the protocol, through e.g. password mismanagement. The adversary can thus dynamically corrupt users, allowing us to capture *user forward secrecy.* We then describe **Prot1**, a protocol secure in this sense under the liberal exposure rule. We only show how the basic model is altered.

5.1 The Password-Adaptive Ideal World

Reveal Password, i**.** $\mathcal{M}^*$ specifies user i, and receives pw_i from $\mathcal{RM}^*$. We shall say that *user i's password has been revealed.* Also, from now on we say that i's password is *known* if it has been revealed or successfully guessed.
Transcript: $(\text{"}reveal\ \ password\text{"}, i, pw_i)$

A Modification of the Rules for Exposing. When a "start session" operation is performed on (i,j) with $role_{ij} = open$ (resp., $connect$), (i,j) may be exposed through $(0,k)$ if $(0,k)$ has been initialized with $PIDS_{0k} = (ID_i, PID_{ij})$ (resp., $(0,k)$'s exchange has been completed with $PIDS_{0k} = (PID_{ij}, ID_i)$), and either PID_{ij} is not the identity of an initialized user, or PID_{ij} is the identity of a user whose password is known, or pw_i is known. As in section 4, this is **liberal** exposure. To get **conservative** exposure, remove the condition "or pw_i is known".

5.2 The Password-Adaptive Real World

Reveal Password, i**.** $\mathcal{M}$ specifies a user i and receives pw_i. The terminology introduced above carries over to the real world.
Transcript: $(\text{"}reveal\ \ password\text{"}, i, pw_i)$

5.3 Prot1

Let G be a group of prime order q, g be a generator of G, $Enc := (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, and $Sig := (\mathcal{K}_S, \mathcal{S}, \mathcal{V})$ be a public-key signature scheme. Run $\mathcal{K}_E(1^\eta)$ to get (pk_E, sk_E) and $\mathcal{K}_S(1^\eta)$ to get (pk_S, sk_S). Set $sk_{\mathcal{T}} :=$

(sk_E, sk_S) and $pk_{\mathcal{T}} := (pk_E, pk_S)$. Finally, let $\{H_n\}_n$ be a family of universal hash functions, mapping into $\{0,1\}^{2\ell}$ for $\ell \in \mathbb{N}$. (q and ℓ depend on η.) **Prot1** runs as follows. "Randomly" means "uniformly at random".

1) $\mathcal{A}$ chooses exponent $x \in \mathbb{Z}_q$ randomly, computes $X \leftarrow g^x$, computes ciphertext $c_{\mathcal{A}} \leftarrow \mathcal{E}_{pk_E}(X, pw_{\mathcal{A}}, ID_{\mathcal{A}}, ID_{\mathcal{B}})$, and sends $c_{\mathcal{A}}$ to $\mathcal{T}$;

2) $\mathcal{T}$ decrypts $c_{\mathcal{A}}$ and checks $\mathcal{A}$'s password. It chooses a hash index n randomly, computes $\sigma_{\mathcal{T}1} \leftarrow \mathcal{S}_{sk_S}(X, n, ID_{\mathcal{A}}, ID_{\mathcal{B}})$, and sends $(X, n, ID_{\mathcal{A}}, ID_{\mathcal{B}}, \sigma_{\mathcal{T}1})$ to $\mathcal{B}$;

3) $\mathcal{B}$ verifies the signature. It chooses y randomly, computes $Y \leftarrow g^y$, computes $c_{\mathcal{B}} \leftarrow \mathcal{E}_{pk_E}(X, Y, n, pw_{\mathcal{B}}, ID_{\mathcal{A}}, ID_{\mathcal{B}})$, and sends $c_{\mathcal{B}}$ to $\mathcal{T}$;

4) $\mathcal{T}$ decrypts $c_{\mathcal{B}}$ and checks $\mathcal{B}$'s password and the values X and n. $\mathcal{T}$ computes $\sigma_{\mathcal{T}2} \leftarrow \mathcal{S}_{sk_S}(X, Y, n, ID_{\mathcal{A}}, ID_{\mathcal{B}})$, and sends $(Y, n, ID_{\mathcal{A}}, ID_{\mathcal{B}}, \sigma_{\mathcal{T}2})$ to $\mathcal{A}$;

5) $\mathcal{A}$ verifies the signature. It computes master key $MK_{\mathcal{A}} \leftarrow H_n(Y^x)$, and then parses it into two equal-length bitstrings $SK_{\mathcal{A}}$ and $\kappa_{\mathcal{A}}$. $SK_{\mathcal{A}}$ is the session key, and $\kappa_{\mathcal{A}}$ is the confirmation code. $\kappa_{\mathcal{A}}$ is sent to $\mathcal{B}$;

6) $\mathcal{B}$ computes master key $MK_{\mathcal{B}} \leftarrow H_n(X^y)$, parses it into two strings SK and $\kappa_{\mathcal{B}}$ of length ℓ. If $\kappa_{\mathcal{B}} \neq \kappa_{\mathcal{A}}$, $\mathcal{B}$ stops. Otherwise, $\mathcal{B}$ sets $SK_{\mathcal{B}} \leftarrow SK$.

Theorem 1. *If Enc is IND-CCA-2-secure, Sig is EU-ACMA-secure, and the DDH assumption holds in G,* ***Prot1*** *is secure under liberal exposures against password-adaptive network adversaries.*

Definitions of IND-CCA-2-security, EU-ACMA-security, and the DDH assumption, and an idea of how the proof is carried out are in the appendix. We show how **Prot1** fails against KCI.

One example involves $\mathcal{A}$, $\mathcal{B}$, and two server instances, $(\mathcal{T}, 1)$ and $(\mathcal{T}, 2)$. Steps 1) to 4) are done between $\mathcal{A}$, $\mathcal{B}$, and $(\mathcal{T}, 1)$. $\mathcal{A}$ has chosen $X = g^x$, $(\mathcal{T}, 1)$ has chosen n_1, $\mathcal{B}$ has chosen $Y = g^y$, and a message with (Y, n_1) is on its way to $\mathcal{A}$, and caught by $\mathcal{M}$. $\mathcal{M}$ replays to $(\mathcal{T}, 2)$ the first message $\mathcal{A}$ sent, so $(\mathcal{T}, 2)$ chooses n_2. Next, $\mathcal{M}$ uses $pw_{\mathcal{B}}$ to compute a third message c encrypting (X, V, n_2), where V is an element that depends on X and Y_1. c is sent to $(\mathcal{T}, 2)$, which computes a fourth message that is sent to $\mathcal{A}$, which computes $SK_{\mathcal{A}}||\kappa_{\mathcal{A}} \leftarrow H_{n_2}(V^x)$. If $\mathcal{A}$ starts using, and $\mathcal{M}$ gets, $SK_{\mathcal{A}}$, $\mathcal{M}$ gets $H_{n_2}(V^x)$. $\mathcal{B}$ is then at risk because the assumptions made on G and $\{H_n\}_n$ do not imply that $H_{n_1}(g^{xy})$ is incomputable in this case since V *is a function of* X *and* Y. Thus $\mathcal{M}$ could compute the code $\mathcal{B}$ expects. In the end, $\mathcal{M}$ does impersonate $\mathcal{A}$ to $\mathcal{B}$, but $\mathcal{M}$ has to work a lot beyond knowing $pw_{\mathcal{B}}$. Remember, revealing $pw_{\mathcal{B}}$ is sufficient for liberal exposures of $\mathcal{B}$ *in the ideal model*; this does not mean that it is sufficient in practice.

Slight variations of this attack are the only KCIs possible against **Prot1** so it seems reasonable to state that **Prot1** does heavily mitigate KCI.

6 Password-and-State-Adaptive Network Adversaries

We further enhance the model by granting the adversary the power to corrupt user instances in the following sense: upon corruption, a user instance's *Unerased*

Internal State (UIS) is revealed. This models storage of ephemeral data in insecure memory (see [28]). Revealing internal state is not often considered for PAKEs since it leads to immediate OffDAs. This is not the case here precisely because we have server private keys. It is however crucial that certain random bits be erased as soon as they have served their purpose.

Our goal is this: if a user instance's UIS *alone* is revealed, that instance's session key, **and at most one other instance's session key**, is compromised.

Section 6.3 describes **Prot2**, which is provably secure in this model. **Prot1** is insecure in this sense. (But see section 6.4.)

6.1 The Password-and-State-Adaptive Ideal World

Corrupt Instance, (i, j)**.** (i, j) should not be terminated and should not have started a session. If (i, j) has been the target of this operation, we shall simply say that (i, j) *has been corrupted.* This terminology applies in the real world (described below) as well. During the execution, an instance that has been corrupted is either *unbound* or *bound.* Upon corruption, it starts out unbound.

Transcript: $\big(\text{"}\mathit{corrupt}\ \text{"}, (i, j)\big)$

A Further Modification of the Rules for Exposing. We first allow (i, j) to be exposed if it has been corrupted. This is the *relaxed exposure rule.* If (i, j) was already connected using the special connection rule defined below, it remains so.

Next, if there exist $(0, k)$ and (i', j') such that (i, j), (i', j'), and $(0, k)$ have matching roles and partners, (i, j) may be exposed if (i', j') is corrupted and unbound. (i', j') then becomes bound. This is the *special exposure rule.*

A Special Premature Connecting Rule. Let (i, j) have the role *connect*, and let (i', j') and $(0, k)$ be such that (i', j') is open for connection from (i, j) through $(0, k)$. If (i, j) is corrupted and unbound, we allow the adversary to prematurely connect (i, j) to (i', j') through $(0, k)$. In this case, $\mathcal{M}^*$ receives $SK_{i'j'}$, and (i, j) becomes bound. This is the *special connection rule.*

The mechanism enforcing the fact that at most one additional instance is compromised solely due to a corruption is the attribution of the "bound" or "unbound" status to a corrupted instance. Intuitively, if in the course of a simulation a user instance must absolutely be exposed but its natural partner instance is already bound, the logic of the protocol should imply that a relevant password is known to the adversary, thereby allowing an ordinary exposure to take place.

6.2 The Password-and-State-Adaptive Real World

Corrupt Instance, (i, j)**.** $\mathcal{M}$ specifies an initialized and not yet terminated user instance, and recovers all unerased internal state of that instance.

Transcript: $\big(\text{"}\mathit{corrupt}\text{"}, (i, j)\big)$ and $\big(\text{"}\mathit{impl}\text{"}, \text{"}\mathit{internal\ \ state}\text{"}, (i, j), \mathit{IntSt}_{ij}\big)$

Revisiting Dictionary Attacks. The comments made in section 4.2 still apply here. Ideally, corrupting user instances has no effect on an adversary's ability to test password guesses beyond the one or two online impersonations that can be performed on a server instance. This accurately captures the idea that dictionary attacks are not aided by UIS revealing. But this is just a definition; it does not show how a protocol concretely achieves this... (See the next paragraph.)

6.3 Prot2

The setup is the same as for **Prot1**, except that the H_n map into $\{0,1\}^\ell$, and the server selects nonces of some length depending on η. **Prot2** runs as follows.

1) $\mathcal{A}$ selects x randomly, computes $X \leftarrow g^x$, and sends $(X, ID_\mathcal{A}, ID_\mathcal{B})$ to $\mathcal{T}$;

2) $\mathcal{T}$ selects a nonce N randomly, and sends $(X, N, ID_\mathcal{A}, ID_\mathcal{B})$ to $\mathcal{B}$;

3) $\mathcal{B}$ selects y and n randomly, computes $Y \leftarrow g^y$ and $MK_\mathcal{B} \leftarrow H_n(X^y)$, ***erases*** y, and computes $c_\mathcal{B} \leftarrow \mathcal{E}_{pk_E}(1, X, Y, n, N, pw_\mathcal{B}, ID_\mathcal{A}, ID_\mathcal{B})$. ***It erases the randomness used to compute*** $c_\mathcal{B}$, and sends $c_\mathcal{B}$ to $\mathcal{T}$;

4) $\mathcal{T}$ decrypts $c_\mathcal{B}$ and checks $pw_\mathcal{B}$ and the random values N and X. It computes $\sigma_{\mathcal{T}1} \leftarrow \mathcal{S}_{sk_S}(1, X, Y, n, N, ID_\mathcal{A}, ID_\mathcal{B})$, and sends $(Y, n, N, \sigma_{\mathcal{T}1})$ to $\mathcal{A}$;

5) $\mathcal{A}$ checks the signature $\sigma_{\mathcal{T}1}$. It computes $SK_\mathcal{A} \leftarrow H_n(Y^x)$, ***erases*** x, and computes $c_\mathcal{A} \leftarrow \mathcal{E}_{pk_E}(2, X, Y, n, N, pw_\mathcal{A}, ID_\mathcal{A}, ID_\mathcal{B})$. ***It erases the randomness used to compute*** $c_\mathcal{A}$ and sends $c_\mathcal{A}$ to $\mathcal{T}$;

6) $\mathcal{T}$ decrypts $c_\mathcal{A}$, checks $pw_\mathcal{A}$, and checks the random values X, Y, n, and N. It computes $\sigma_{\mathcal{T}2} \leftarrow \mathcal{S}_{sk_S}(2, X, Y, n, N, ID_\mathcal{A}, ID_\mathcal{B})$, and sends $\sigma_{\mathcal{T}2}$ to $\mathcal{B}$;

7) $\mathcal{B}$ checks the signature $\sigma_{\mathcal{T}2}$. It sets $SK_\mathcal{B} \leftarrow MK_\mathcal{B}$.

Theorem 2. *If Enc, Sig, and G are as in theorem 1,* ***Prot2*** *is secure under conservative exposures against password-and-state-adaptive network adversaries.*

Proof ideas are in the appendix. Since the theorem holds under the conservative exposure rule, **Prot2** fully avoids KCI. What about UIS revealing?

Revisiting Dictionary Attacks *Again*. UIS is basically "whatever information is needed for an instance to continue its computations". Everything else – as specified in the protocol description above – is *erased* as soon as it is no longer needed. In particular, what (say) $\mathcal{A}$ needs to hold on to after sending its first protocol message is its exponent x, otherwise it cannot compute the session key. What it does not need to hold on to is the randomness used in the public-key encryption performed to compute the fifth protocol message in step 5). Hence this randomness is erased and therefore off limits to the UIS revealing. But the fact that the encryption is randomized is exactly what protects the password, even if the other parts of the plaintext are known to the attacker. This is the *very definition* of semantic security.

It is worth contrasting this with the situation in CRS-based 2-PAKEs that are standard-model-secure and that use smooth projective hashing. Such protocols also protect the password with public-key encryption. Recall that decryption in this setting is impossible; nobody even has the decryption key. Now, this problem is circumvented using the smooth projective hashing mechanism. But this is where the catch lies in terms of internal state: for the mechanism to work, the parties *need to hold on to the randomness used for encryption, and it becomes UIS.* Thus, revealing UIS causes trivial dictionary attacks.

Replaying Stale Data. As for the security goal that we aim to achieve with UIS revealing, it is indeed reached by this protocol intuitively because the random group elements of both instances are bound to both passwords. In particular, if an adversary who gets the exponent x of an originator wants to successfully replay $X = g^x$ in a protocol run, it can only do so through knowledge of the originator's password because the fifth protocol message contains an encryption of the new responder's fresh group element. A stale message cannot therefore simply be replayed.

6.4 Unerased Internal State in Prot1

We return briefly to **Prot1** to see what happens to it under UIS revealing.

Note that it certainly does not fulfill the security goal, for if the exponent x chosen by $\mathcal{A}$ is revealed to adversary $\mathcal{M}$, $\mathcal{M}$ can replay the first protocol message to many new server instances communicating with many other $\mathcal{B}$ instances and compute the correct session key and confirmation code each time, without $\mathcal{M}$ ever needing to reveal $pw_{\mathcal{A}}$ or $pw_{\mathcal{B}}$.

However, notice that even though the desired goal defined in case of UIS revealing is not achieved, UIS revealing *still fails to open the protocol to dictionary attacks*, for the same reason as in the case of **Prot2**: no reasonable definition of UIS for **Prot1** will contain the encryption's random bits.

7 Conclusions and Future Work

We made a (theoretical and practical) case for considering 3-PAKE[spk]s. We proposed a hierarchy of security models to accommodate them, and exhibited protocols that fit in (parts of) this hierarchy. In the process, we were able to make what we believe to be a sensible definition of security against internal state revealing that does not trivially break password security.

Several directions should be explored from here. First, the models could be extended to accommodate arbitrary password distributions. It has already been argued for instance in [8] and [11] that simulation-based security models are well suited to do this. Second, the models can be enhanced to capture server forward secrecy, in the sense that compromise of the server's long-term secret key should not damage past-established session keys. It is highly likely that **Prot1** and **Prot2** can be proven secure in this sense. Third, considering UC-security (see [11]) for 3-PAKE[spk]s would be fruitful. Fourth, it may be possible to consider queries that reveal *erased state* by demanding that a subset of random bits remains hidden. Admittedly, practical scenarios which would lead to this kind of attack may be somewhat rare, but it is still worth exploring in an effort to continue refining provable security for KE.

Acknowledgments. The present project is supported by the National Research Fund, Luxembourg. The authors wish to thank Dalia Khader, Peter Y. A. Ryan, and Qiang Tang for fruitful discussions on this topic, and the anonymous reviewers for their helpful insight.

References

1. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth Projective Hashing for Conditionally Extractable Commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009)
2. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-Based Authenticated Key Exchange in the Three-Party Setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
3. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In: STOC 1998, pp. 419–428 (1998)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
5. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Bellovin, S., Merrit, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: Proceedings of the IEEE Symposium on Research in Security and Privacy (1992)
7. Boyarsky, M.: Public-Key Cryptography and Password Protocols: the Multi-User Case. In: 6th ACM Conf. on Computer and Communications Security (CCS), pp. 63–72 (1999)
8. Boyko, V., MacKenzie, P., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
9. Byun, J.W., Jeong, I.R., Lee, D.H., Park, C.-S.: Password-Authenticated Key Exchange between Clients with Different Passwords. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 134–146. Springer, Heidelberg (2002)
10. Camenisch, J., Casati, N., Gross, T., Shoup, V.: Credential Authenticated Identification and Key Exchange. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 255–276. Springer, Heidelberg (2010)
11. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
12. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
13. Gennaro, R.: Faster and Shorter Password-Authenticated Key Exchange. In: ACM Conference on Computer and Communications Security (2008)
14. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)

15. Goldreich, O., Lindell, Y.: Session-key Generation Using Human Passwords Only. J. Cryptology 19(3), 241–340 (2006)
16. Goyal, V., Jain, A., Ostrovsky, R.: Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
17. Groce, A., Katz, J.: A New Framework for Efficient Password-Based Authenticated Key Exchange. In: 17th ACM Conf. on Computer and Communications Security (CCS), pp. 516–525. ACM Press (2010)
18. Halevi, S., Krawczyk, H.: Public-Key Cryptography and Password Protocols. In: 5th ACM Conf. on Computer and Communications Security (CCS), pp. 122–131 (1998)
19. Hao, F., Ryan, P.Y.A.: Password Authenticated Key Exchange by Juggling. In: Christianson, B., Malcolm, J.A., Matyas, V., Roe, M. (eds.) Security Protocols 2008. LNCS, vol. 6615, pp. 159–171. Springer, Heidelberg (2011)
20. Jablon, D.: Strong Password-Only Authenticated Key Exchange. ACM Computer Communications Review 26(5), 5–26 (1996)
21. Jiang, S., Gong, G.: Password Based Key Exchange with Mutual Authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
22. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
23. Katz, J., Vaikuntanathan, V.: Round-Optimal Password-Based Authenticated Key Exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011)
24. Lucks, S.: Open Key Exchange: How to Defeat Dictionary Attacks without Encrypting Public Keys. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 79–90. Springer, Heidelberg (1998)
25. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
26. Lin, C.-L., Sun, H.-M., Hwang, T.: Three-Party Encrypted Key Exchange: Attacks and a Solution. ACM Operating Systems Review 34(4), 12–20 (2000)
27. Lin, C.-L., Sun, H.-M., Steiner, M., Hwang, T.: Three-party Encrypted Key Exchange Without Server Public-Keys. IEEE Communications Letters 5(12), 497–499 (2001)
28. Shoup, V.: On Formal Models for Secure Key Exchange. IBM Research Report RZ 3120 (April 1999)
29. Steiner, M., Tsudik, G., Waidner, M.: Refinement and Extension of Encrypted Key Exchange. ACM Operating Systems Review 29(3), 22–30 (1995)
30. Wang, F., Zhang, Y.: A New Security Model for Cross-Realm C2C-PAKE Protocol. IACR e-print archive (2007)
31. Wu, S., Zhu, Y.: Client-to-Client Password-Based Authenticated Key Establishment in a Cross-Realm Setting. Journal of Networks 4(7) (2009)
32. Yeh, H.-T., Sun, H.-M., Hwang, T.: Efficient Three-Party Authentication and Key Agreement Protocols Resistant to Password Guessing Attacks. Journal of Information Science and Engineering 19(6), 1059–1070 (2003)

A Computational Assumptions and Security Notions

DDH and Entropy Smoothing. Let g generate group G of prime order q. The *Decisional Diffie-Hellman* (DDH) assumption states that given (g, g^x, g^y), g^{xy} and g^z are computationally indistinguishable when x, y, and z are chosen randomly. Combining it with the *entropy smoothing* of hash family $\{H_n\}_n$, the assumption becomes: given (g, g^x, g^y, n), $H_n(g^{xy})$ and R are computationally indistinguishable, where x, y, hash index n, and $R \in \{0,1\}^\ell$ are chosen randomly.

IND-CCA-2-Secure Encryption. A public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *indistinguishable encryptions against adaptive chosen-ciphertext attacks* if every probabilistic polynomial-time (PPT) adversary $\mathcal{M}$ has negligible (in η) advantage in the following game. $\mathcal{M}$ receives pk as input, where $(pk, sk) \leftarrow \mathcal{K}(1^\eta)$. $\mathcal{M}$ first makes *decryption queries* on ciphertexts of its choice: it outputs any string c and receives in return $m \leftarrow \mathcal{D}_{sk}(c)$. Next it outputs two equal-length messages (m_0, m_1). Then, $b \in \{0,1\}$ is chosen randomly, and $\mathcal{M}$ receives $c' \leftarrow \mathcal{E}_{pk}(m_b)$. $\mathcal{M}$ now makes more decryption queries, excluding the challenge c'. Finally, $\mathcal{M}$ outputs a bit $\hat{b}$. $\mathcal{M}$'s *advantage* is $\left|\mathbb{P}[\hat{b} = b] - \frac{1}{2}\right|$.

EU-ACMA-Secure Signing. A public-key signature scheme $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ has *existentially unforgeable signatures against adaptive chosen-message attacks* if every PPT adversary $\mathcal{M}$ has negligible advantage in the following game. $\mathcal{M}$ receives pk as input for $(pk, sk) \leftarrow \mathcal{K}(1^\eta)$. It then makes *signature queries* on messages of its choice: it outputs any message m and receives $\sigma \leftarrow \mathcal{S}_{sk}(m)$. Finally, $\mathcal{M}$ outputs a pair (m', σ'). $\mathcal{M}$'s advantage is $\mathbb{P}\big[(m', \sigma') \leftarrow \mathcal{M}(pk)\big]$, where σ' is a valid signature on m' and m' was never the input to a signature query.

B Indications on Security Proofs

$\mathcal{M}^*$ is built from $\mathcal{M}$ by having $\mathcal{M}^*$ play the role of $\mathcal{RM}$ for $\mathcal{M}$, using its interactions with $\mathcal{RM}^*$ to answer $\mathcal{M}$'s requests. We give indications on: 1) how passwords are treated and the exact role the encryption scheme plays and 2) how real-world message deliveries are translated into ideal-world events.

Dealing with Passwords. First, $\mathcal{M}^*$ needs to answer message deliveries *without the users' passwords*, because $\mathcal{RM}^*$ assigns passwords outside of $\mathcal{M}^*$'s view. To get around this, whenever a user encrypts a message, $\mathcal{M}^*$ encrypts a dummy password instead. By the encryption's security, $\mathcal{M}$ cannot tell the difference.

Next, $\mathcal{M}^*$ keeps track of which messages are from users, and which are from $\mathcal{M}$, so $\mathcal{M}^*$ can then tell which of these should be translated into password guesses. For example, if $\mathcal{M}$ has a message from $\mathcal{M}$ delivered to a server expecting something from user i, $\mathcal{M}^*$ first decrypts this message and obtains a password pw. $\mathcal{M}^*$ then asks $\mathcal{RM}^*$ if $pw = pw_i$, and answers the message delivery accordingly. Notice that $\mathcal{M}^*$ learns pw_i if and only if $\mathcal{M}$ does.

Third, we indicate why encryption must be IND-CCA-2-secure. It is well-known that this is equivalent to it being non-malleable (under adaptive attacks), and non-malleability is required here because encryption does not just hide the password: it also *non-malleably binds it to the users' random choices.*

Connection Assignments and UIS Revealing. Here is a sample of the reasoning used for theorem 2. Instances that output the first message get the role *open* since they compute the key first. The others get the role *connect*. Let (i,j) have the role *open* and suppose it accepts a fourth message. By the security of the signature and the uniqueness of the nonce N, a unique $(0,k)$ with $PIDS_{0k} = (ID_i, PID_{ij})$ has output a fourth message on input $(1, X, V, m, N, ID_{ij}, PID_{ij})$, where X was chosen for (i,j), and V and m are a group element and hash index. $(0,k)$ must also have received (i,j)'s first message. Suppose $PID_{ij} = ID_{i'}$ for some user i'. If the third message received by $(0,k)$ was honestly generated by an instance, it is unique, and of the form (i',j') for $PID_{i'j'} = ID_i$. Also, V and m were correctly chosen for (i',j'). If neither (i,j) nor (i',j') has been corrupted, $\mathcal{M}^*$ asks $\mathcal{RM}^*$ to start (i,j)'s session, and (i,j) is open for connection from (i',j'). $\mathcal{RM}^*$ assigns to (i,j) a session key SK_{ij} chosen randomly. If $\mathcal{M}$ asks to reveal SK_{ij}, $\mathcal{M}^*$ forwards this request to $\mathcal{RM}^*$, and SK_{ij} is returned to $\mathcal{M}$. Suppose now that $\mathcal{M}$ corrupts (i',j') before it receives any sixth message. $\mathcal{M}^*$ cannot respond to this with $H_m(X^v)$ (where $v = log_g(V)$), as it is inconsistent with SK_{ij}. So, (i',j') connects to (i,j) with the *special connection rule*, $\mathcal{RM}^*$ hands $SK_{i'j'} \leftarrow SK_{ij}$ to $\mathcal{M}^*$, and $\mathcal{M}^*$ gives $MK_{i'j'} \leftarrow SK_{i'j'}$ to $\mathcal{M}$.

This is just a sample. Complete and rigorous proofs (which are very lengthy for both theorems) are in the full version of the paper.

Towards a More Secure and Scalable Verifying PKI of eMRTD

Nicolas Buchmann and Harald Baier

da/sec Biometrics and Internet Security Research Group
Hochschule Darmstadt, Darmstadt, Germany
{nicolas.buchmann,harald.baier}@h-da.de

Abstract The new electronic passport stores biometric data on a contactless readable chip to uniquely link the travel document to its holder. This sensitive data is protected by a complex protocol called Extended Access Control (EAC) against unlawful readouts. EAC is manifold and thus needs a complex public key infrastructure (PKI). Additionally EAC is known to suffer from unsolved weaknesses, e.g., stolen (mobile) passport inspection systems due to its missing revocation mechanism. The paper at hand seeks for potential approaches to solve these shortcomings. As a result we present an evaluation framework with special focus on security and scalability to assess the different candidates and to give a best recommendation. Instead of creating new protocols, we focus on solutions, which are based on well-known protocols from the Internet domain like the Network Time Protocol (NTP), the Online Certificate Status Protocol (OCSP), and the Server-based Certificate Validation Protocol (SCVP). These protocols are openly standardised, widely deployed, thoroughly tested, and interoperable. Our recommendation is that the EAC PKI would benefit most from introducing NTP and OCSP.

1 Introduction

Travel documents have become a natural part of travelling into foreign countries for citizens. The best known and most frequently used travel document is the passport. According to [7] the passport is the basic official document, which denotes a person's identity and citizenship and provides an assurance for the state of transit or destination that the holder can return to the state of issuance. For international operability passports are specified by the International Civil Aviation Organisation (ICAO), who sets the standards for all its contracting states.

1.1 Context

The ePassport is a new passport with capabilities for biometric identification with the help of a contactless integrated circuit chip. It is intended to achieve a new level of travel document security, by a strong bond between the electronic Machine Readable Travel Document (eMRTD) and its holder [7]. Passengers

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 102–118, 2014.

will, besides the security advantages, profit from faster border control clearance times, because of the possible automation of the border control. However the storage of biometric data and the wireless interface of the chip lead to security concerns, e.g., the leakage of personal data. The European Union (EU) addresses these worries by implementing an additional security protocol called *Extended Access Control* (EAC) [3], which aims at protecting the citizen's sensitive data by only granting access to authorised parties as defined by the issuing country.

1.2 Motivation

EAC is on the one hand very powerful and can satisfy nearly all security and privacy demands. On the other hand, however, it requires a very complex public key infrastructure (PKI) to ensure the authenticity and authorisation of inspection systems [6, 8]. We call this public key infrastructure the *Verifying PKI.* As of today the Verifying PKI does not have a mechanism for revocation. To limit the value of a stolen inspection system the issued inspection system certificates have a very short validity period, usually only one day. This creates an enormous effort with respect to both generation of key pairs/certificates and their distribution, because currently every inspection system needs a new certificate everyday, for each country that issues an EAC enabled eMRTD.

In addition to the complexity of the key generation and certificate distribution, an attacker may abuse a stolen inspection system for two main reasons: first, the certificate is valid for its entire validity period and cannot be revoked. Second the eMRTD does not have an accurate time source. Thus in general the eMRTD is not able to detect an expired certificate, which extends the possible attack period. Introducing a reliable revocation system would not only make EAC more secure, but also would lower the burden for creating certificates with extremely short validity periods.

The complexity of the Verifying PKI and the obvious inadequacies of the EU EAC protocol are the starting point of the paper at hand. Finding and evaluating possible solutions for the shortcomings of EAC are the main goals of this paper. Although EAC and its Verifying PKI is used within the European Union, there is no discussion about the aforementioned drawbacks in the standardisation documents. Although the scientific community identified some weaknesses and came up with solution candidates (e.g., [4, 6, 15, 17]) there is no evaluation of the candidates and thus no best recommendation. Additionally if a solution is given, it is self-created instead of using well-established standards.

1.3 Results and Contributions

Our first contribution is to extend the discussion of [6,15] and to highlight the two main shortcomings of the EU EAC, namely the missing revocation mechanism of inspection systems and the absence of an accurate time source of an eMRTD. We then discuss the resulting potential risks. In order to strengthen EAC we present solution candidates. In addition to previously proposed enhancements of the community (e.g., [6]), we introduce two promising approaches: first, as an authentic and precise time source the Network Time Protocol (NTP, [13]) is proposed together with the Online Certificate Status Protocol (OCSP, [9]) as lightweight revocation mechanism. Second the Server-based Certificate Validation Protocol (SCVP, [11]) is assessed, which transfers the certificate chain validation, the time acquirement, and the revocation check to an external server. In contrast to existing proposals using these proven Internet standards is also a design choice with the future in mind, because the next generation of smart card technology will be able to directly communicate via TCP/IP [18]. As with all new technologies this feature will at first be solely available in high-end smartcards. In compliance with Moore's law TCP/IP will in the near future become available in cheap mass-produced bulk smartcards. This can be justified simply because TCP/IP is widely adopted in a variety of other domains and will provide a huge gain of comfort and interoperability.

In order to come up with our best recommendation for an actual practical use, we develop an assessment methodology with a special focus on security, scalability, and standardisation. We evaluate the candidates against our weighted criteria. We also evaluate our proposal against the most promising, self-created approach from the current scientific community, the Hoepman protocol [6] The final outcome is that the Verifying PKI would benefit most from introducing NTP and OCSP. Assuming an online connection of the inspection system with a sufficient bandwidth is reasonable due to the pervasiveness of UMTS/LTE. Hence even mobile terminals may implement our solution.

1.4 Organisation of This Paper

The rest of the paper is organised as follows: Section 2 summarises related contributions, places our work in the context of these papers, and demonstrates the gap in existing research, which is filled by our work. After presenting the related work, Section 3 describes the status quo of the security infrastructure of eMRTDs. Then Section 4 explains the shortcomings of the current security protocols and presents potential solutions of these weaknesses. We develop our weighted assessment methodology in Section 5, evaluate the solution candidates against our list of criteria, and compare the results with the Hoepman protocol. Finally, Section 6 concludes our paper and points to future work.

2 Related Work

This section summarises related contributions, places our work in the context of these papers, and demonstrates the gap in existing research, which is filled by our work.

Moses [15] gives in his white paper from Entrust a comprehensive view on the weaknesses of the current Verifying PKI and proposes a workaround. Instead of revoking the certificates and providing a real-time clock, he proposes to compensate this deficiency with strong confidential storage and restriction of using the reader's private key to authorised operators, e.g., due to a storage of the private key in the back office. His self-assessment of this solution is 'brittle, because there is no way to recover when it goes wrong'. In contrast our solution also works if the terminal's private key has already been compromised. [15] states that the absence of a real-time clock makes revocation ineffective. We agree with this statement and will only evaluate solutions that provide a real-time clock and revocation.

Hoepman et al. [6] present weaknesses and propose security improvements for a variety of eMRTD protocols. Relevant for our paper are only the proposed improvements for EAC. Hoepman et al. [6] sketch an idea of a self-invented online terminal authentication (Hoepman protocol) and define certain boundary conditions (e.g., resistance to replay attacks). The proposed protocol is actually very similar, to one of our proposals, the SCVP protocol (see Section 4.2). It also delegates the actual terminal authentication to a trusted third-party. However, in contrast to our SCVP proposal the Hoepman protocol separates the terminal access rights from the terminal certificates. Although this is a promising approach, which provides real-time revocation, it suffers from two drawbacks: first, there is no detailed specification of the mechanism. Second it is a new protocol, which has to be investigated thoroughly. Our solution candidates, however, are based on well-known and well-established Internet standard protocols that have been proven useful in other domains for a long time. Additionally Hoepman et al. [6] do not provide an assessment methodology to evaluate solution candidates. Nevertheless we will evaluate the Hoepman online terminal authentication , because in our opinion it is currently the most promising, self-created approach from the scientific community.

Vaudenay and Vuagnoux [21] report the weaknesses of EAC and describe certain attack scenarios, but do not propose improvements for EAC.

Chaabouni and Vaudenay [4] introduce the idea to have identity checks when leaving a domestic country to have more frequent clock updates. To provide certificate revocation they propose a reputation-based trust mechanism where a threshold authentication proof is created by a collaboration of a certain number of neighbour terminals. The proposed additional identity check does indeed shorten the possible attack period, but it does not completely solve the problem, because during a long vacation an eMRTD's date is still not up-to-date. A reputation-based revocation system solves the problem of a single stolen terminal, but the authors present no detailed analysis how to integrate such a revocation system in the eMRTD infrastructure. An attacker still has the option to steal a sufficient number of terminals and compromise them to exceed the threshold for the authentication proof. We propose solutions that provide a real-time date and revocation independent of the number of stolen terminals by an attacker.

Pasupathinathan et al. [17] present a self-made protocol called On-Line Secure E-Passport Protocol (OSEP Protocol). The authors claim to solve weaknesses of EAC. The OSEP protocol drops the access control flexibility of EAC and a terminal sends private information from the eMRTD to the country's embassy. In contrast to Pasupathinathan et al. [17] we think that both facts raise privacy concerns: a terminal, which needs access to the document holder's name stored on the chip, should not automatically get access to the sensitive fingerprints. Furthermore countries, which may track travellers through their embassies, is a show-stopper for any travelling privacy. Comparable to [6] we do not think that another self-made protocol is needed where no practical experience data exists how the protocol performs in practice. We also do not consider it realistic that the EU will drop EAC, because of a practically untested protocol. With these characteristics, the OSEP protocol disqualifies itself and we will not evaluate it against our proposals and the Hoepman protocol.

3 Verifying Public Key Infrastructure

This section introduces the purpose and hierarchy of the Verifying PKI. It has its own distinctive purpose. Without an additional mechanism there is no limitation who is allowed to read out the chip's data, if a third party gets hold of the physical document, regardless whether the document was handed over willingly, or an attacker obtained physical possession without the bearer's approval. To limit the access to the sensitive biometric data only to selected authorities, a PKI is needed to grant and validate the access rights of the inspection systems, the so-called Verifying PKI. It is well described in [19].

The evaluation of the access rights and validation of the authenticity from the terminal certificates has to be done by the eMRTD chip itself. Therefore the Verifying PKI uses card-verifiable certificates [14] to speed up the process. This process is called Terminal Authentication which is part of EAC.

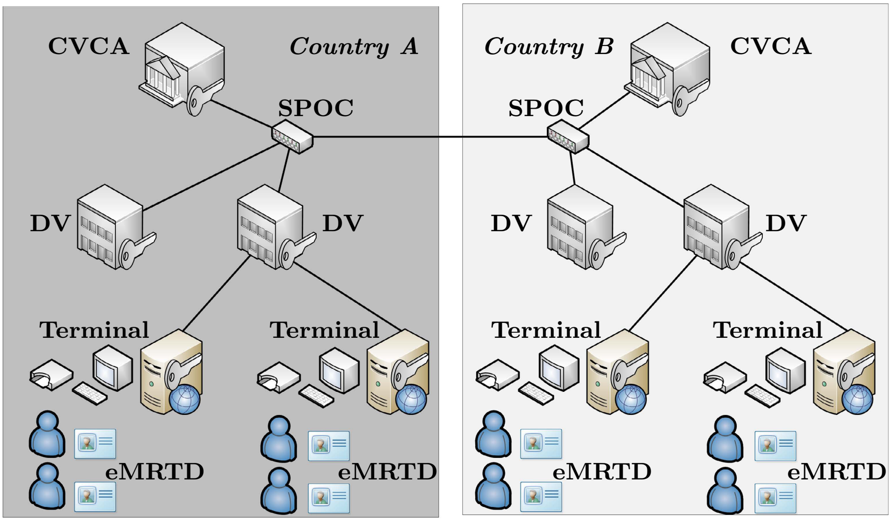

Fig. 1. The Verifying PKI and SPOC

The Verifying PKI structure can be seen in Fig. 1 on the facing page. The hierarchy of the PKI is as follows. Every country has a Country Verifying Certificate Authority (CVCA) which holds a self-signed certificate. The CVCA's task is to certify Document Verifiers (DV) by signing their certificates with the CVCA Private Key. A CVCA does not only sign domestic DV certificates, but also foreign DV ones to grant access for other member states to the eMRTD issued by the CVCA's state. Therefore a DV needs one certificate for each country that issues an eMRTD with EAC. There can be multiple DVs for every country.

As trust point the CVCA certificate of the issuing country is stored on every eMRTD chip with EAC support. Below the DV hierarchy level is the level of the terminals with their terminal certificates also called inspection system certificates. Terminals which need access to the sensitive data of eMRTDs from different countries need a certificate for each country granted by their DV.

For issuing the terminal certificates the DV does not need to contact every country itself, but instead every country maintains a so-called Single Point of Contact (SPOC), which is responsible for transnational communications and cross-border certifications. Each country that issues an eMRTD with EAC support or wants to read an eMRTD needs its own SPOC.

4 Shortcomings of Verifying PKI and Solution Candidates

In Section 4.1 we first address the shortcomings of the Verifying PKI, i.e. the missing revocation mechanism and the missing time source for an eMRTD. Once we have identified the drawbacks we discuss in Section 4.2 solution candidates to improve the Verifying PKI.

4.1 Shortcomings of the Current EU EAC Implementation

Although the new protocols specified by the EU EAC standard [3] are sophisticated and thus enhance the security of former protocols, the current EAC standard still offers two unsolved weaknesses. They are linked to the new Verifying PKI and the associated Terminal Authentication. First, the eMRTD has no access to a precise and authentic time source, so it can not accurately validate if a terminal certificate is still valid. Instead a pseudo clock mechanism is used, which is described below. The second problem is that a certificate once issued stays valid until the expiration date no matter what happens.

As announced above the missing time source is replaced by a pseudo clock mechanism, which works as follows:

The eMRTD stores a date T_{eMRTD} in an internal register which gets updated during the Terminal Authentication. Initially T_{eMRTD} is set during the chip personalization to the personalization date.

During the Terminal Authentication the eMRTD reads the "Expiration date" $T_{Cert,Expiration}$ field from all certificates and validates that $T_{Cert,Expiration}$ is not before T_{eMRTD}.

After every successful Terminal Authentication the eMRTD chip reads all certificates from the chain and checks which got the latest "Effective Date" $T_{Cert,Effective}$ field, which is the equivalent of the "Not Before" field from X.509 certificates. If $T_{Cert,Effective} > T_{eMRTD}$ then T_{eMRTD} is set to $T_{Cert,Effective}$ and stored in the internal register [3] [19].

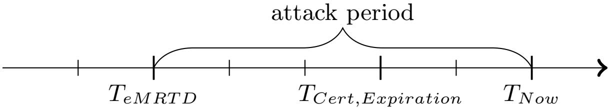

Fig. 2. Timeframe for attack

The problem which arises is that the eMRTD chip can only detect a past $T_{Cert,Expiration}$ value if it is used often, because then T_{eMRTD} is relatively accurate. Nevertheless an eMRTD chip can not be sure if a terminal certificate is really still valid, because if $T_{eMRTD} \leq T_{Cert,Expiration} < T_{Now}$ the eMRTD will not detect an expired certificate. The possible attack period is shown in Fig. 2. So for a successful attack the attacker must already have a valid certificate and he can expand the time period in which his terminal can read biometric data. This only works if the eMRTDs chip gets no accurate time update from another terminal.

Possible reasons why the eMRTD got no clock might be explained from the problems that arise if you try to integrate a clock into an eMRTD, e.g. the missing power source.

4.2 Solution Candidates to Introduce Revocation

As of today, the Verifying PKI neither supports CRLs nor OCSP [2]. Although the reason is not given in [2] we believe that it is due to the missing external connection of the eMRTD and memory restrictions on the eMRTD. Computing power increases through progress and the next generation of smart cards will support TCP/IP, therefore these are no more obstacles [18].

Common Certificate Revocation Lists (CRL) [10] will not be discussed, because they can grow arbitrary large after some time [20,23] and parsing through a big revocation list might also take unnecessary long for the eMRTD chip [23]. Their overhead makes them unattractive for low-power devices as the eMRTD.

In the following subsections we present potential solution candidates to introduce a revocation mechanism to the Verifying PKI and discuss the practical feasibility, respectively. Our starting point are protocols, which are standardised for the Internet domain and thus thoroughly investigated.

OCSP. The Online Certificate Status Protocol (OCSP) provides an alternative to CRLs [9]. Besides real-time status OCSP provides the benefit of lower bandwidth usage per request and no storage requirement compared to a CRL. A drawback is that the OCSP responder has to be available all the time [23].

Using OCSP as revocation mechanism in the Verifying PKI delivers many benefits over traditional CRLs. The eMRTD chip does not need additional internal memory for storing the list, the download size is smaller and simultaneously the eMRTD chip does not need to process an entire list of CRL entries. The eMRTD chip can get a direct response if the certificate has been revoked or is valid depending on its regular expiration date. Even so [4] has a different focus it favours OCSP to improve EAC. A time source is still needed, because the OCSP responder does not check the validity period of the certificate, but instead if it has been revoked and with the OCSP extension CertHash [1] also if the certificate has really been issued by the CVCA. In our use case the validity period of the certificates will be verified by the eMRTD as part of Terminal Authentication and the OCSP responder should only send the status "good" or "revoked". The "unknown" status is prohibited.

SCVP. The Server-Based Certificate Validation Protocol (SCVP) in contrast to OCSP provides a server based full validation of a certificate, with optional revocation [11]. The complete certificate path creation, validation and check for revocation is done by an SCVP server. If the client trusts the server it can delegate nearly the complete PKI overhead to the SCVP server. This enables the use of a PKI for low-end devices. To check the validity status of a certificate the SCVP server uses either CRLs or OCSP. SCVP is not widely used yet, but has been tested on smart cards [16]. SCVP provides authenticity and integrity of the request and response messages, but does not ensure confidentiality. However, the SCVP standard suggests to use the Transport Layer Security Protocol (TLS) if confidentiality is needed [12].

An eMRTD chip with support for the SCVP would not only benefit from the features of OCSP, but would also no longer depend on a clock. SCVP messages would be signed by the CVCA (or a dedicated SCVP service) and SCVP also provides measures against replay attacks. Despite the need for a transport protocol between inspection system and eMRTD chip, because of the missing Internet connection, SCVP seems to be the best solution in view of benefit and created effort at a first glance.

4.3 A First Conclusion

For a revocation it is mandatory that the eMRTD chip can securely communicate with a trusted home server. This is not possible without extra infrastructure to handle the requests and the willingness of the inspection system to play the role as a network bridge between the Internet and the current simple smart card communication protocols. Due to the availability of UMTS or LTE this assumption even holds for mobile inspection systems. Mobile inspection systems without Internet access can fall back to the validation of the physical security features, face recognition or a manual validation with the picture printed on the data page. OCSP and SCVP both provide good solutions if the infrastructure obstacles (i.e., the high availability demands) can be handled. Both protocols effectively solve the problem of stolen terminals and their efficiency has been proven in other domains. The classical CRL is not suitable for the EAC revocation, because of the low-power eMRTD chip.

5 Evaluation of the Solution Candidates

This section introduces an evaluation scheme and applies it to the two solution candidates from Section 4. After the evaluation the candidates are evaluated against the Hoepman protocol [6] in Section 5.8. The two solution candidates NTP together with OCSP (from now on referred to as NTP+OCSP) and SCVP both theoretically provide a solution to the given requirements, but can be evaluated differently against the following criteria. The criteria are mostly based on the well-known Software Engineering non-functional requirements [22].

Table 1. Evaluation Ratings

criterion	NTP+OCSP	SCVP
Security (replay & man-in-the-middle attack)	+	+
Convenience & Acceptability (border check time, privacy)	○	+
TCO (hardware, software, reusability)	○	-
Scalability (network load, home server load)	+	○
Reliability & Availability (complexity, points of failure)	○	○
Feasibility (economical)	+	-

Security is the first criterion which the candidates are evaluated against. This criterion consists of the resistance against certain attacks like replay attacks, and man-in-the-middle attacks.

Convenience and Acceptability are the next criteria which reflect the end user's benefits and drawbacks which the respective solution provides.

The *Total Cost of Ownership (TCO)* is a criterion which not only depends on the new technology needed to provide the services, but also on the reusability of existing IT structures.

If another country wants to introduce a new technology and therefore the global number of users changes dramatically, then *Scalability* is the criterion which considers this.

The fifth criteria are *Reliability and Availability* which rate the dependence on other systems and if the systems are loosely coupled or if they heavily rely on other components.

Feasibility is the last criterion which also includes how likely it is that a certain technology will be integrated into an eMRTD.

Some criteria are not independent e.g. the scalability can influence the availability and therefore the user's acceptance and so on.

In the eMRTD domain we consider *Security* and *Scalability* the most important criteria. On the one hand security is an absolute must, because of the embedded biometric data and on the other hand scalability is very important, because in tourist seasons the passengers boarding airplanes can increase drastically and smooth operation of inspection systems must still be ensured. Therefore *Security* and *Scalability* will weight double for the final score.

For every criterion the candidates are rated positive (+), neutral (○) or negative (-) and for the final rank the individual ratings get points, these are then summed up to receive an end result.

5.1 Security

In this section both candidates are evaluated for potential weaknesses against common security attacks like replay attacks, and man-in-the-middle attacks.

The first item is the resistance against replay attacks. NTP, OCSP and SCVP all provide nonce support to individually link the unique request/response pairs, by default or via a protocol extension. The lightweight OCSP profile [5] should not be used, because it removes the nonce in favour of better scalability which is achieved by response pre-production and response message caching. To prevent replay attacks unique request/response pairs are essential.

Next topic is the resistance against man-in-the-middle attacks. NTP, OCSP and SCVP support the use of digital signatures for authenticity and therefore prevent man-in-the-middle attacks. Independent of the supported mechanisms, there is no direct use case for a man-in-the-middle attack, because confidentiality is no security goal, due to the fact that all time information and the revocation status are not considered confidential. A possible attack would be an Impersonation Attack in which the attacker tries to make the client believe that he is a legitimate server. This attack is also prevented by the same mechanisms as the man-in-the-middle attack.

A general security concern might be the introduction of TCP/IP itself, because it might open new attack vectors to the eMRTD. This can easily be mitigated by only allowing a one to one connection between the inspection system and the eMRTD with exactly one open socket. So with this careful design decision both protocols will provide a higher security increase than potential TCP/IP flaws a security decrease. TCP/IP is needed by OCSP and SCVP. Therefore this will not influence the rating.

From a security point the two candidates have no significant weaknesses. On the one hand the fact that NTP+OCSP consist of two different protocols whose services must be provided by two different daemons, even if they are running on the same server, provide two potential weak points and SCVP only one. On the other hand could the independence of both services also be treated positive, because it might be harder for an attacker to disturb both services. So this depends on the actual implementation and should not influence the rating. Therefore both candidates get a positive security rating.

5.2 Convenience and Acceptability

The user's convenience directly influences the acceptance of a certain technology. So a criterion must be how the new protocols influence the average border check time. A main benefit from the new protocols is better data privacy for the biometric data stored on the eMRTD.

The solution candidate's influence, on the border check processing time, shall be the first item for evaluation. The NTP+OCSP solution has the disadvantage that it can only lengthen the eMRTD evaluation, because the EAC verifying card-verifiable certificate chain must still be validated by the chip and the additional steps for time acquisition and the certificate revocation cost additional time irrespective of how much. The SCVP solution can make the evaluation process shorter, require the same amount of time or even could take longer.

For SCVP the certificate chain validation itself will take a shorter time, because the SCVP server has more computation power than a small smart card microprocessor. Two new potential time additions come to the verifying process on the SCVP server compared to current verification on the eMRTD. These are the acquisition of an accurate time and the certificate revocation mechanism. Both can be done independent of the certificate verification if NTP and automatic CRL download is used by the server. If OCSP is used by the server it would negatively influence the validation time and therefore a CRL should be preferred. We expect the SCVP validation process to be faster than on the chip and the only variable remaining is the transmission of the request and response.

Calculating an exact transmission time is not possible, because it depends on at least the bandwidth and the distance to the home SCVP server.

The next item of consideration shall be how the data privacy benefits from the solution candidates. NTP+OCSP and SCVP both provide the same benefit that expired terminal certificates will always be rejected and that stolen or compromised terminal certificates can be revoked effectively. Both mechanisms provide the same benefit, but one question is if the protocols could leak private information or enable tracking of the document holder. Neither NTP, OCSP nor SCVP send travel document specific data to the home server. NTP does not send any privacy relevant data at all and OCSP/SCVP send only data identifying the inspection system to the home server. Therefore the only negligible privacy concern is that the home server's operator could learn that one of the country's million passports is currently presented to the terminal. The operator is not able to identify the document holder any further and therefore we do not consider this a privacy risk.

So all protocols only provide a benefit and pose no risk to the document holder's data privacy.

Both solution candidates can provide convenience for the users and therefore boost their acceptance. NTP+OCSP provides all the benefits that SCVP does, but can only slow down the border check handling therefore it gets a neutral rating and SCVP a positive rating.

5.3 Total Cost of Ownership (TCO)

This chapter focuses on the expense necessary for the solution candidates. First the necessary new hard- and software will be assessed. Furthermore it is important which components of the already existent system can be reused or integrated directly or indirectly for example after a firmware update.

NTP, OCSP and SCVP have a relatively equal impact on the eMRTD PKI and inspection system structure. The eMRTD chip is not upgradeable via a firmware update, so only the next generation of eMRTDs could support the new protocols. If the current chip is powerful enough to perform all three protocols is hard to tell, but all of them have already been implemented on a regular smart card [16]. The current eMRTD chip is powerful enough to validate card-verifiable certificate chains, so it should be powerful enough to handle a time stamp package and an OCSP response or an SCVP response. Also neither of the protocols need any additional persistent storage space. So the financial impact on the eMRTD itself should be minimal from a hardware perspective. The software has to be changed of course to support the new protocols.

The next items to evaluate are the changes necessary to the inspection systems. The necessary modifications for the inspection systems operating system should be patchable with a new firmware. So only development costs occur, but no hardware upgrade costs. For NTP+OCSP and SCVP an Internet connection is necessary to communicate with the respective home server. The inspection system must already communicate with its DV and this DV must communicate with its country's SPOC. So some sort of network connection should already be present. Upgrading the broadband connection for the inspection system might be necessary as well as an upgrade for the SPOC to handle real time requests.

The last item for potential upgrade costs is at the home server. NTP+OCSP and SVCP need some sort of home server for every issuing country with a connection to the country's SPOC. The server must provide an NTP server and an OCSP responder or an SCVP server. To provide authenticity and integrity all three protocols support the use of digital signatures. The generation of the signatures could be accelerated by using Hardware Security Modules (HSM). Standard CPUs are also needed to handle the protocol request and the certificate chain creation and revocation for OCSP.

Even without exact figures SCVP and NTP+OCSP can be compared. On the one hand NTP+OCSP cost two HSM runs for digital signature generation because they are two stand-alone protocols and SCVP only one, but on the other hand SCVP needs more CPU time for the certificate chain building, revocation and verification, than NTP+OCSP for a revocation and system clock lookup.

The bandwidth consumption of NTP+OCSP and SCVP should be minimal for both. NTP+OCSP might have higher development costs for the eMRTD chips software, the inspection system software and the SPOC's software. The development costs should be minimal compared to the required hardware costs for the home server. As already mentioned above, NTP+OCSP might require more HSM signing runs and less CPU power, then SCVP. For the actual NTP+OCSP specification it could be considered to drop the internal signatures and instead sign both responses together in one big response block. With such an optimisation only the CPU time remains, which is much higher per SCVP request than per OCSP request. So NTP+OCSP gets a neutral ranking and SCVP a negative ranking for the TCO, because of the higher CPU time costs.

5.4 Scalability

Scalability describes the system's behaviour if the requirements on supported user clients change drastically. The increased input can influence the performance, because of higher resource requirements which depend on the complexity of the entire system. One criterion to evaluate is the load per request, which directly influences the system's scalability. The load on the home server and on the network between inspection system and home server can be differentiated.

To compare the network load of NTP+OCSP with the one from SCVP it must be taken into consideration that the protocols will most likely be implemented in a more lightweight form. The NTP network impact is minimal and therefore only OCSP and SCVP shall be compared. All certificates must be checked for revocation in case of OCSP. In case of SCVP all certificates need verification. The requests could contain all necessary certificates or just the serial numbers of the certificates which would result in a lower network usage.

For OCSP the serial number is always enough, because even if the OCSP responder does not know the associated certificate for the serial number, the certificate revocation status is considered good.

For SCVP a serial number certificate look up must always provide a result, because otherwise no verification of the complete chain is possible. The SCVP server can be easily provided with the CVCA certificate because it is present in the same country. The DV certificate's signing requests are all handled by the SPOC which shall also be connected to the SCVP server. Therefore an automatic supply of DV certificates should also be possible without requiring major effort. One problem however lies in the acquisition of the terminal certificates. They are created by the DV, for every terminal, on a daily basis and the serial number remains unknown for the SCVP server. So for SCVP the terminal certificate must be sent entirely instead of just the serial number.

SCVP would have a higher average bandwidth usage than NTP+OCSP. For the TCO scoring, it was already estimated that SCVP would have a higher CPU load per request. Therefore NTP+OCSP gets a positive rating and SCVP a neutral one.

5.5 Reliability and Availability

Reliability and the linked availability are influenced by the solution candidates complexity and the resulting points of failure. A terminal not supporting the protocols or even a broken terminal always breaks the regular border control procedure and is out of scope for this evaluation. NTP+OCSP and SCVP need an Internet connection to communicate with the home server. If the connection fails, all three protocols will not work. They also need the verifying country and issuing country SPOC to be online at all times. Both are points of failure as well as the home server of the issuing country. On the home server runs the NTP and OCSP daemon or the SCVP daemon to process the requests from the eMRTD. All of these are potential points of failure.

One small difference here is that for NTP+OCSP two daemons could stop working and for SCVP only one, but again the purpose of NTP and OCSP is independent, so one service still running from two could also be considered as a better circumstance than a complete breakdown of a single service.

NTP+OCSP and SCVP have no big difference in their points of failure. It could be argued that the tasks of SCVP are more complicated and more prone to error, but this would involve potential implementation errors which are out of scope. Both candidates heavily rely on external systems and therefore both get a neutral rating.

5.6 Feasibility

Feasibility for the solution candidates can be divided into technical feasibility, financial feasibility, economical feasibility and the basic conditions concerning the existing infrastructure.

From a technical perspective all solution candidates are possible. NTP, OSCP and SCVP were implemented for some research projects on smart cards and can therefore be considered as technical feasible on the eMRTD chip.

The financial part was already evaluated in, Section 5.3 therefore this shall not have an impact on the feasibility evaluation. The economical feasibility shall be the matter at hand. NTP+OCSP and SCVP both extend the European EAC mechanism and provide effectively the same benefit. Financial factors aside both candidates require a certain amount of development effort. NTP+OCSP are two protocols, but do not automatically lead to the doubled development effort, because the protocols are older, simpler and most likely more common to the developers for the implementation on a smart card. What sets the difference is that NTP and OCSP could be more or less directly implemented on a smart card with little or no development effort for the home server. SCVP in the eMRTD would need an implementation with a single request response pair and the missing access to the terminal certificates for the home server would enlarge the request or require more effort for the DVs. That is why NTP+OCSP are considered more likely with this simple analysis than SCVP.

SCVP needs a more complicated home server, the protocol would have to be adjusted and would create more burden for the DVs. Therefore NTP+OCSP gets a positive rating and SCVP a negative rating.

5.7 Evaluation Result

Table 1 on page 110 shows a summary of the solution candidates evaluation results and Table 2 on the following page presents the final results. The positive rating gets two points, the neutral rating one point and the negative rating

Table 2. Points and Result

criterion	NTP+OCSP	SCVP
Security (x2)	4P	4P
Convenience & Acceptability	1P	2P
TCO	1P	0P
Scalability (x2)	4P	2P
Reliability & Availability	1P	1P
Feasibility	2P	0P
Point Sum	13P	9P
Final Rank	1	2

zero points. Additionally the points for *Security* and *Scalability* will be doubled. NTP+OCSP ranks first with 13 points and SCVP second with 9 points. So the recommended solution by our evaluation is NTP+OCSP.

5.8 Hoepman Protocol Ranking

In this section we evaluate the Hoepman protocol [6] (see Section 2) against our solution candidates.

Our evaluation favoured NTP+OCSP over SCVP, which is very similar to the Hoepman protocol. So to get a picture if the Hoepman protocol is more attractive than our winner, we assess if there are criteria in which it is stronger than SCVP, and if this difference is sufficient to rank better than NTP+OCSP.

For the *Security* criterion the Hoepman protocol only defines resistance to replay attacks as a boundary condition, but this should not be a practical problem, because comparable to SCVP this could be achieved with a unique request/response pair by nonce support.

The results for *Convenience & Acceptability* and *TCO* are also comparable to those of SCVP.

What makes the difference between the Hoepman protocol and SCVP are the criteria *Scalability, Reliability & Availability, and Feasibility*, because it is a new, self-made protocol without any practical experience from use in other domains. Therefore it is unrealistic to be as optimized as SCVP, a well-established Internet standard, for scalability, reliability, and availability. Also the dynamic access control is theoretically a nice feature, but it makes the home server even more complicated, which again badly influences scalability and the feasibility of introducing this mechanism.

In conclusion with the current specification [6] the Hoepman protocol will perform worse than SCVP, and therefore is in any case less attractive for the eMRTD domain, than NTP+OCSP.

6 Conclusion and Future Work

This paper presented weaknesses and possible solutions for the current EU EAC implementation for eMRTDs. The problems of the missing accurate time and the linked terminal certificate revocation were explained and shown how they can be solved. A winner was found in the form of NTP with OCSP. [4] also favours OCSP and states that an authentication proof involving the home CVCA would be the ultimate trust mechanism for EAC. Nevertheless a great amount of new infrastructure would be needed for NTP, OCSP or SCVP.

For future work both solutions could be optimized to become more attractive by merging NTP and OCSP into one protocol without violating the standards, or striping possible unnecessary overhead from SCVP.

Another topic for future discussion is the creation of an OCSP extension to carry the terminal access rights instead of encoding them into the terminal certificates. Either to override the access rights of a valid certificate or to completely outsource the access rights to the real-time OCSP. Which would be similar to the Hoepman protocol [6], but based on proven Internet standards.

For a look in the future to see if our solution might become reality in a fourth generation of eMRTDs it depends on the specific future requirements of EAC, and if the effort for the terminal certificate revocation and the precise validation time can be justified.

Acknowledgment. This work was supported by the European Commission through the FIDELITY EU-FP7 project (Grant No. SEC-2011-284862) and CASED.

References

1. Bickenbach, H.J., Brauckmann, J., Alfred, G., Horváth, T., Knobloch, H.J.: Common PKI Specifications for interoperable Applications, v2.0 (January 2009), http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf
2. BSI: Certificate Policy für die ePass-Anwendung der hoheitlichen Dokumente. Bundesamt für Sicherheit in der Informationstechnik (BSI) (October 2010)
3. BSI: Technical Guideline TR-03110 Advanced Security Mechanisms for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Bundesamt für Sicherheit in der Informationstechnik (BSI), 2.05 edn. (October 2010)
4. Chaabouni, R., Vaudenay, S.: The extended access control for machine readable travel documents, EPFL (February 2010)
5. Deacon, A., Hurst, R.: The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments. RFC 5019 (Proposed Standard) (September 2007)
6. Hoepman, J.-H., Hubbers, E., Jacobs, B., Oostdijk, M., Schreur, R.W.: Crossing borders: Security and privacy issues of the european e-passport. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 152–167. Springer, Heidelberg (2006)

7. ICAO: ICAO MRTD Report, vol. 1(1) (March 2006)
8. ICAO: ICAO MRTD Report, vol. 2(1) (April 2007)
9. IETF: RFC 2560 – X.509 Internet Public Key Infrastructure – Online Certificate Status Protocol - OCSP
10. IETF: RFC 3820 – Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile
11. IETF: RFC 5055 – Server-Based Certificate Validation Protocol (SCVP)
12. IETF: RFC 5246 – The Transport Layer Security (TLS) Protocol - Version 1.2
13. IETF: RFC 5905 – Network Time Protocol Version 4: Protocol and Algorithms Specification
14. ISO: Identification cards – Integrated circuit cards – Part 8: Commands for security operations – ISO/IEC 7816-8 (September 2009)
15. Moses, T.: Protecting biometric data with extended access control – securing biometric datasets in electronic identification documents, Entrust, Inc. (January 2010), http://download.entrust.com/resources/download.cfm/23504/
16. Papapanagiotou, K., Markantonakis, C., Zhang, Q., Sirett, W.G., Mayes, K.: On the Performance of Certificate Revocation Protocols Based on a Java Card Certificate Client Implementation. In: Sasaki, R., Qing, S., Okamoto, E., Yoshiura, H. (eds.) SEC 2005. IFIP AICT, vol. 181, pp. 551–564. Springer, Heidelberg (2005)
17. Pasupathinathan, V., Pieprzyk, J., Wang, H.: An on-line secure e-passport protocol. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 14–28. Springer, Heidelberg (2008)
18. Rankl, W., Effing, W.: Smart Card Handbook, 4th edn. Wiley (August 2010)
19. Straub, T., Hartl, M., Ruppert, M.: Digitale Reisepässe in Deutschland - Prozesse und Sicherheitsinfrastruktur. In: Dittmann, J. (ed.) Sicherheit. LNI, vol. 77, pp. 233–243. GI (2006)
20. van Tilborg, H.C., Jajodia, S. (eds.): Encyclopedia of Cryptography and Security, 2nd edn. Springer (September 2011)
21. Vaudenay, S., Vuagnoux, M.: About machine-readable travel documents. Journal of Physics: Conference Series 77(1) (2007)
22. Wiegers, K.E.: Software Requirements, 2nd edn. (Pro-Best Practices). Microsoft Press (March 2003)
23. Yum, D.H., Lee, P.J.: Separable Implicit Certificate Revocation. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 121–136. Springer, Heidelberg (2005)

Mutual Restricted Identification*

Lucjan Hanzlik, Kamil Kluczniak, Mirosław Kutyłowski, and Łukasz Krzywiecki

Faculty of Fundamental Problems of Technology,
Wrocław University of Technology
firstname.secondname@pwr.wroc.pl

Abstract. We extend the idea of Restricted Identification deployed in the personal identity documents in Germany. Our protocol, Mutual Restricted Authentication (MRI for short), is designed for direct anonymous authentication between users who belong to the same domain (called also a sector). MRI requires *only one* private key per user. Still there are no limitations to which domain a user may belong and the domains are not fixed in advance. This enables an implementation of MRI when a strictly limited secure memory is available (like for smart cards). MRI guarantees that a user has exactly one identity within a domain, while the identities from different domains of the same user are not linkable. The main difference between RI and MRI is that for MRI the privacy of both participants are protected, while in case of RI the terminal is fully exposed. The protocol is efficient, extremely simple (in particular, it outperforms RI) and well suited for an implementation on resource limited devices such as smart cards.

Keywords: personal ID document, Restricted Identification, privacy, simultability, authentication, AKE.

1 Introduction

In pervasive systems one of the key issues is identifying and authenticating digital artefacts. This concerns all electronic identity documents but also other devices like smartphones, tablets, and identification tokens. So we have to talk about an *electronic-ID* (e-ID for short). In some cases the same e-ID has to play different roles in different subsystems – called from now on *domains* – and use a different identity in each domain. Unless necessary, an *e-ID* device should not use linkable identities in different domains. E.g., professional and private roles should be strictly separated.

For technical and usability reasons wireless communication will play a dominant role for communication with e-ID devices. So protecting information exchange against eavesdroppers becomes a key issue. Information on membership of, and identity in, a domain should also be protected. Moreover, tough rules on personal data protection and social sensitivity in countries like UK and Germany make it necessary to *guarantee* effective protection.

Today, most e-ID systems do not hide the identity of at least one party. This is the case for *machine readable travel documents* (that is, electronic passports and personal

[*] This work is based on business and security analysis done in project O ROB 0027 01/ID 27/2 financed by NCBiR.

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 119–133, 2014.

identity documents) and so the terminals *must* be trusted. However, if e-ID devices wish to interact directly, privacy of both sides should be protected.

Domains and Restricted Identification. The idea of separating activity areas was implemented first in the Austrian *Bürgerkarte* - this system is based on the passwords computed with a symmetric algorithm from the citizen's personal number.

The next step was development of the *nPA*, the new German personal identity document. Restricted Identification (RI for short) protocol [1], allows an nPA to use a single private key to authenticate against any terminal. Each nPA uses its private key to compute its domain specific identifiers. The key feature of RI is unlinkability: two terminals from two different domains cannot determine if they are interacting with the same nPA or with two different nPA's. However, within a single domain all actions of an nPA *must* be attributed to the same anonymous identity.

The protocol from nPA requires a prior execution of the Terminal Authentication (TA) protocol, during which the terminal signs with its private key a nonce provided by the nPA. Thereby the transcript of communication can be used as an *undeniable proof* of interaction with the terminal. Therefore, this protocol is not suitable for the case of peer-to-peer communication between e-IDs.

Design Goals. In this paper we develop Mutual Restricted Identification protocol (MRI) that expands RI [1].

MRI is fully *simultable*, i.e. each side of the protocol can compute a transcript of communication that is indistinguishable from the transcripts obtained from real communications. This resolves the problem stated above, no participant can use a communication transcript as a proof against a third party. MRI provides *unlinkability* for activities in different domains, just as in case of RI. MRI is *symmetric* regarding operations performed by both sides of the protocol. This feature has a positive impact on implementation costs and flexibility. MRI is is *resilient to leakages* – in many scenarios revealing ephemeral keys does not disclose the session keys (which is not true for nPA). This also concerns forward security: revealing long-term secrets does not reveal the session keys. MRI is slightly more *efficient* than RI. Therefore, it is well suited feasible for smart cards implementation (which has been also confirmed by an implementation on Java Cards).

Previous Work. There are many papers on authenticated key exchange (AKE). The AKE protocols secure in the Canetti-Krawczyk (CK) model [2], guarantee that the adversary cannot distinguish established keys from random values, as long as some session secrets (ephemeral keys) are not leaked. In [3] Krawczyk proposed a variant of CK and proved that the HMQV protocol (a hashed version of MQV from [4]) achieves so called weak perfect forward security (wPFS), resilience to key compromise impersonation (KCI) attacks and revealing the ephemeral keys of a single party. The extended Canetti-Krawczyk model (eCK) was proposed in [5] to capture combinations of static and ephemeral keys corruptions (apart from the obvious ones that break security by definition), including revealing both ephemeral keys or both static keys. NAXOS [5], NAXOS+ [6], and CMQV [7] were shown to be secure in eCK model.

The KEA+ protocol [8] was shown to be secure in a model weaker that eCK that allows revealing the long-term key of at most one of the parties.

In the above mentioned protocols each party has prior knowledge on the ID of the other party, or the identifiers are sent during a protocol execution. The later case may lead to privacy violations, thus identity hiding was concerned in the papers [9,10,11,12]. Deniability, as an additional feature was achieved in the PACE|AA protocol [13]. In this protocol each party can create transcripts of protocol runs with the same probability distributions as for the transcripts coming from the real protocol executions. Deniability of SKEME and partial deniability of SIGMA were discussed in [14].

From the above mentioned protocols based on DH key exchange (without pairings) none fully satisfied the required goals:

- the following protocols are not deniable: NAXOS, NAXOS+, JFKi, JFKr, SIGMA,
- the following protocols are not identity hiding: MQV,HMQV,CMQV,
- the following protocols use prior knowledge of the partner's ID: KEA+, NAXOS, NAXOS+, SKEME.

On the other hand the protocols [10] and [11] are based on pairings, and it is not clear how could they be adjusted for restricted identification.

The Restricted Identification protocol has its variant called ChARI, which redefines initial steps and eliminates so called *group keys* shared by many e-IDs. The price paid is a slight loss of efficiency and the use of separate certificates, whitelists or blacklists for domains. Below we present an efficiency comparison for RI, ChARI and MRI:

Table 1. Efficiency comparison for RI protocols

protocol	exponentiations on a smart card	exponentiations on terminal	communication rounds	number of private keys on a smart card
RI [1]	2 + 2	2 + 1	3	2
ChARI [15]	2 + 2	3 + 1	3	1
MRI (this paper)	3	3	2	1

2 Mutual Restricted Identification

Below we use a cyclic group $\mathcal{G}$ of a prime order q where the Discrete Logarithm Problem is hard.

Domains. Two users can authenticate themselves if they belong to the same *domain*. On the other hand, a user may belong to any number of domains. For a domain S there is a uniquely defined generator $g_S \in \mathcal{G}$ used by all users. g_S must be derived in a way that the discrete logarithm of g_{S_1} with respect to g_{S_2} is unknown for any domains S_1, S_2, $S_1 \neq S_2$. For instance, we can use a hash function mapping the legal names to $\mathcal{G}$, that is, $g_S = H(S)$.

Table 2. Mutual Restricted Identification protocol

Alice		Bob
x_A - private key $y_A = g^{x_A}$ - public key $cert_A$ - certificate for y_A		x_B - private key $y_B = g^{x_B}$ - public key $cert_B$ - certificate for y_B
	OPTIONAL SETUP	
recompute g $y_A := g^{x_A}$ - set public key fetch $cert_A$ and check y_A		recompute g $y_B := g^{x_B}$ - set public key fetch $cert_B$ and check y_B
	MAIN PROCEDURE	
choose a at random		choose b at random
$h_A := H(a\|0)$		$h_B := H(b\|0)$
$c_A := y_A^{h_A}$	$\xrightarrow{c_A}$	$c_B := y_B^{h_B}$
	$\xleftarrow{c_B}$	
$K := {c_B}^{x_A h_A}$		$K := {c_A}^{x_B h_B}$
$K_A := H(K\|1)$, $K_B := H(K\|2)$	$\xrightarrow{Enc_{K_A}(a, cert_A)}$	$K_A := H(K\|1)$, $K_B := H(K\|2)$
		reject if $c_A \neq y_A^{H(a\|0)}$ or $cert_A$ invalid
reject if $c_B \neq y_B^{H(b\|0)}$ or $cert_B$ invalid	$\xleftarrow{Enc_{K_B}(b, cert_B)}$	
$K_s := H(K\|3)$		$K_s := H(K\|3)$

Initialization. The protocol is described on Fig. 2. Note that some initial steps are omitted: such as negotiating the encoding format, the communication parameters, the algorithms and group used, etc. This stage must be based on a temporal ad hoc identity and there must be a very limited number of behavior profiles during this phase in order to eliminate identification.

In the following description we assume that the communication is within domain S with the generator $g_S = g$. The certificates for the public keys of, respectively, Alice and Bob in the domain S will be denoted by $cert_A$ and $cert_B$.

Protocol Idea. The first part of the protocol is deriving the master session key K by the Diffie-Hellman protocol based on the values c_A and c_B. At this stage the identities of the participants are not revealed. At the first look it may appear that derivations of K depend on the participants' identities. However, c_A and c_B are in fact equal to $g^{x_A h_A}$ and $g^{x_B h_B}$, and as h_A and h_b are in some sense "random", so are $x_A h_A$ and $x_B h_B$ modulo q.

Note that the key K depends on the domain parameter $g = g_S$. Indeed, if A uses g and B uses a different key g', then A derives $(c_B)^{x_A h_A} = (g')^{x_B h_B x_A h_A}$, while B derives $(c_A)^{x_B h_B} = g^{x_A h_A x_B h_B}$. So the results are different.

The master key K is used to get a number of keys by applying a hash function with different parameters. We follow a frequent practice to yield "independent" keys by hashing a shared secret expanded with different parameters.

The second stage of the protocol is communicating the values a and b. The purpose is the following: knowing the session key by A is an evidence of knowledge of the discrete logarithm of c_A with respect to g. So, as A knows the discrete logarithm of c_A with respect to y_A, we conclude that A may easily derive the discrete logarithm of y_A with respect to g. Thereby after terminating the protocol execution in an accepting state we may conclude that A knows the secret key x_A.

Note that while c_A is "random", finding a such that $c_A = y_A^{h_A}$ is infeasible, if c_A has not been computed in this way. Indeed, possibility of deriving h_A would break the Discrete Logarithm Problem. Namely, we would challenge the adversary with $c_A = g^r$ for r chosen at random, get back h such that $y_A^h = c_A$, and then derive $y_A = g^{r/h}$.

Encrypting a and b has two goals. First, it protects identity information from an eavsdropper. Second, verification is possible only if the recipient knows the decryption key, and therefore has been participating in the whole interaction.

3 Security Assumptions

Definition 1 (DDH Assumption). *Let $\mathcal{G}$ be a cyclic group of a prime order q'. The* Decisional Diffie-Hellman Problem *(DDH Problem) is hard for $\mathcal{G}$ if there is no probabilistic polynomial-time algorithm $\mathcal{A}_{\mathrm{DDH}}$ that with a non-negligible probability distinguishes between the distributions $D_0 = (\tilde{g}, \tilde{g}^{\alpha}, \tilde{g}^{\beta}, \tilde{g}^{\gamma})$ and $D_1 = (\tilde{g}, \tilde{g}^{\alpha}, \tilde{g}^{\beta}, \tilde{g}^{\alpha\beta})$, where α, β, γ are chosen at random from $\{1, \ldots, q' - 1\}$. That is, for any probabilistic polynomial-time algorithm $\mathcal{A}_{\mathrm{DDH}}$ the adversary's advantage*

$$\mathbf{Adv}(\mathcal{A}_{\mathrm{DDH}}) = |\Pr[\mathcal{A}_{\mathrm{DDH}}(D_1) = 1] - \Pr[\mathcal{A}_{\mathrm{DDH}}(D_0) = 1]|$$

is at most ϵ_{DDH} for a negligibly small ϵ_{DDH}.
The Computational Diffie-Hellman Problem *(CDH Problem) is to derive $g^{\alpha\beta}$ given g^{α} and g^{β}. If the DDH Problem is hard, then there is no efficient algorithm solving the CDH Problem.*

In order to model requirements for a hash function we use the notion of correlated-input secure hash functions [16].

Definition 2. *A hash function H is* correlated-input secure *if for a random r and any Boolean circuits $C_1, \ldots, C_n$ there is no adversary such that given $H(C_1(r)), \ldots, H(C_{n-1}(r))$, it distinguishes between $H(C_n(r))$ and a random R of the same length with a non-negligible probability within realistic time.*

For the encryption function we use the Ideal Cipher Model, closely related to Random Oracle Model.

Definition 3. *In the* Ideal Cipher Model *encryption is modelled by an oracle $\mathcal{O}$ that holds a table T storing triples (m, k, c), where m stands for a plaintext, k stands for an encryption key, and c stands for a ciphertext. Initially, T is empty.*

Given a query Encrypt(m,k), the oracle $\mathcal{O}$ checks if there is an entry of the form (m, k, c) in T. If yes, then $\mathcal{O}$ responds with c. Otherwise, $\mathcal{O}$ chooses c' at random, but different from all z such that there is an entry (h, k, z) in T. Then $\mathcal{O}$ responds with c' and inserts (m, k, c') in T.

Given a query Decrypt(c,k), the oracle $\mathcal{O}$ checks if there is an entry of the form (m, k, c) in T. If yes, then $\mathcal{O}$ responds with m. Otherwise $\mathcal{O}$ chooses m' at random, but different from all z such that there is an entry (h, k, z) in T. Then $\mathcal{O}$ responds with m' and inserts (m', k, c) in T.

4 Privacy Issues

Proofs of Interaction. One of the key privacy problems is that a transcript of a protocol can be used by a communicating party or by an eavesdropper to prove that an interaction with a certain party has occurred. This provides motivation to solutions based on the Zero-Knowledge Proof principle, where an interaction can be perfectly simulated and therefore is useless for proving anything. The paper [17] states this property more explicitly as *simultability* of protocol executions.

Proposition 1. *B (respectively, A) can generate a proof consisting of all data transferred during an alleged execution of the MRI protocol together with all internal values used by B (A) without any interaction with A but with exactly the same probability distribution as for the real interactions.*

Proof. B creates a fake transcript by performing all steps on behalf of A and B. The only difference is that B does not attempt to derive K as it is done by A. However, this is unnecessary, since B can compute K using its own procedure.

Creating a fake transcript by A is similar. □

Another possibility is that an eavesdropper holding neither x_A nor x_B presents an interaction transcript. Potentially, it can contain some data that cannot be created without involvement of A or B – in this case we have a proof that either this is a real transcript or a simulated one created by either A or B. However, if we may assume that A and B are honest, then we get a proof of interaction between A and B. Below we show that there is no such a danger for the MRI protocol.

Proposition 2. *In the Ideal Cipher Model under the DDH Assumption, given a transcript of an interaction consisting of c_A, c_B, Enc$_1$ and Enc$_2$, it is infeasible to identify the protocol participants. More precisely, the advantage of the adversary to win the following game is negligible: for arbitrary participants A_0, B_0 and A_1, B_1:*

- *the challenger chooses a bit u at random,*
- *the challenger presents a record T consisting of the messages exchanged between A_u, B_u during a real execution of the MRI protocol,*
- *the adversary responds with u'. He wins if $u' = u$.*

Proof. The original game can be formalized as follows:

Game 0.

choose u, a, b at random
$h_A := H(a|0)$, $h_B := H(b|0)$, $c_A := y_{A_u}^{h_A}$, $c_B := y_{B_u}^{h_B}$
$K := g^{x_{A_u} x_{B_u} h_B h_A}$, $K_A := H(K|1)$, $K_B := H(K|2)$
$E_1 := \mathit{Enc}_{K_{A_u}}(a, \mathit{cert}_{A_u})$, $E_2 := \mathit{Enc}_{K_{B_u}}(b, \mathit{cert}_{B_u})$
$u' := \mathcal{A}(c_A, c_B, E_1, E_2)$

The encryption operations above are understood as calls to the encryption oracle $\mathcal{O}$. Now we replace the encryption results with random variables:

Game 1.

```
    choose $u$, $a$, $b$ at random
    $h_A := H(a|0), h_B := H(b|0), c_A := y_{A_u}^{h_A}, c_B := y_{B_u}^{h_B}$
    $K := g^{x_{A_u} x_{B_u} h_B h_A}, K_A := H(K|1), K_B := H(K|2)$
    choose $E_1$ and $E_2$ at random and inform oracle $\mathcal{O}$ about them
    $u' := \mathcal{A}(c_A, c_B, E_1, E_2)$
```

In Game 1 we simply reverse the order of operations concerning encryption oracle. Instead of asking $\mathcal{O}$ during an encryption we create the values and demand to include these values in the table kept by $\mathcal{O}$. This may lead to conflicts with already existing values and thereby to a fault event. However, this is very unlikely.

In Game 1 the adversary gets 4 values that are uniformly distributed. However, these values are entangled by entries that exist in the table of the encryption oracle $\mathcal{O}$. Disclosing these relations is possible only after asking the oracle $\mathcal{O}$ with the key K_A or K_B. Assume that this is possible with a non negligible probability. We show that then we would be able to solve the DDH Problem. Indeed, for a given instance (g, C, D, Z) we play Game 1 with $y_{A_0} = C$, $y_{B_0} = D$, and observe the queries to $\mathcal{O}$. If any key equals $H(Z^{h_A h_B}|1)$ or $H(Z^{h_A h_B}|2)$, then we have an indication that (C, D, Z) is a Diffie-Hellman triple. □

Passive Adversary and Linking Attempts

Definition 4 (passive adversary privacy model). *We assume that $A_1, \ldots, A_k$ can communicate within domains $S_1, \ldots, S_u$. During the time period considered, the adversary observes t interactions, say $T_1, \ldots T_t$, and participates itself in some number of interactions T (at arbitrary time moments). The adversary knows the participants $A_1, \ldots, A_k$ and their public keys for each domain. An elementary event in the probability space Ω is a mapping that indicates for each transaction the communicating parties and the domain used:*

$$R : \{T_1, \ldots, T_k\} \longrightarrow \{A_1, \ldots, A_k\}^2 \times \{S_1, \ldots, S_u\}$$

A priori knowledge of the adversary is a probability distribution π on Ω.

The probability distribution π on Ω models the knowledge resulting from the real conditions. E.g. if the transmission times of T_i and T_{i+1} overlap, then usually we may conclude that the participants of T_i and T_{i+1} are different.

Definition 5 (attack model for the passive adversary). *Let D be the list of all messages exchanged during some protocol executions observed by the adversary. We consider the distributions π and $\pi|D$ (the probability distribution π conditioned by the data D observed). We say that the protocol is* secure against linking, *if the distributions π and $\pi|D$ do not differ in a non-negligible way. That is, given a sample drawn from distribution π or $\pi|D$, the adversary has no non-negligible advantage to guess whether the sample has been drawn from π or $\pi|D$.*

Definition 5 says that the data sent by the protocol does not add substantial *new* knowledge for determining *who is talking with whom*. Note also that π might be arbitrary, as real conditions and users' behavior is hard to predict. In particular, we are not making the artificial assumption that π is the uniform distribution.

Unlinkability - Sketch of the Proof. The security of the MRI protocol against linking follows from similar considerations as in the proof of Proposition 2. However, now within the game we take into account all interactions, each game concerns choice of participants as well as domain used, and the adversary is given all transcripts.

Before we proceed let us introduce the following concept. For a pair of participants A and B holding the public keys $y_A = g_S^{x_A}$, $y_B = g_S^{x_B}$ for a domain S we define their *hidden public key* as $g_S^{x_A x_B}$.

Proposition 3. *Given the hidden public key $g_S^{x_A x_B}$ for A and B and domain S, one can generate transcripts of an interaction between A and B within S with exactly the same probability distribution as for the real interactions.*

Proof. The fake transcripts are created by following exactly the operations of A and B from the description of the protocol. The only exception is computing the key K (as neither x_A nor x_B is available). However, one can compute K using the equality $K = (g_S^{x_A x_B})^{h_A h_B}$. □

Obviously, ability of the adversary to distinguish between distributions π and $\pi|D$ from Definition 5 can only increase, if for each pair of participants A and B and each domain the adversary learns the hidden public key $g_S^{x_A x_B}$. From now on we assume that the adversary knows the hidden public key for each pair of participants and domain.

Assume that the adversary applies algorithm $\mathcal{A}$ to break privacy. The overall strategy to show that advantage of $\mathcal{A}$ is negligible is as follows:

We consider behavior $\mathcal{A}$ separately for different *cases*. A *case* is determined by fixing the value of R. (Note that according to our assumptions, the probabilities of cases may differ.) However, if we succeed to show that in each case we can replace the transcripts by random transcripts with a negligible change of behavior of $\mathcal{A}$, then $\mathcal{A}$ may skip the input regardless of the case.

Now consider a case C, and assume that the last interaction T_k is between the participants A and B. Then we consider two kinds of inputs to $\mathcal{A}$: the original transcripts and the transcripts with the last interaction T_k replaced by four random messages. As in the proof of Proposition 2 we show that the behavior of $\mathcal{A}$ cannot differ non-negligibly for these two kinds of inputs. Assume conversely that $\mathcal{A}$ behaves in a different way. Then we use it to build a distinguisher between the random transcripts and the transcripts between participants A and B. Indeed, given a transaction T which is either random or between A and B, we build a case for $\mathcal{A}$, by adding transcripts $T_1, \ldots, T_{k-1}$ where the participants of the interactions are indicated by C. Creating the transcripts is possible due to Proposition 3.

We proceed in the same way, in each phase we replace the next T_i by random transcripts and we argue that the behavior of $\mathcal{A}$ cannot change in a non-negligible way. Finally we are left with random transcripts, but $\mathcal{A}$ behaves almost in the same way as for the original inputs for the case C.

Finally notice that after these transformations $\mathcal{A}$ works on the same sets of random inputs with the same probability distribution. Hence $\mathcal{A}$ may skip the actual input and generate random transcripts by itself. It follows directly that $\mathcal{A}$ does not distinguish between π and $\pi|D$. Thereby we get the following result:

Theorem 1. *Assuming the Ideal Cipher Model and hardness of the Decisional Diffie-Hellman Problem, the MRI protocol is secure against linking.*

5 AKE Security of the MRI Protocol

For security of the session key we follow the model originating from [18] and extended by many authors. The model is based on the principle that if one of the legitimate participants (not necessarily both!) enters an *accepting state* with a session key K_s, then it should be infeasible for the adversary to derive K_s. In an accepting state a participant A not only holds the session key but also the identifier of the accepted session and the identity of the other participant B with whom A believes to share K_s.

The adversary $\mathcal{A}$ fully controls the communication channel between any participants A and B. This means that if a message is sent from A to B (or conversely), then $\mathcal{A}$ may prevent the delivery, may modify the message, or deliver a message of its choice. Moreover, $\mathcal{A}$ may deliver a message when no message is sent.

Security Game. We confine ourselves to the case when there are participants A and B holding private keys x_A, x_B. $\mathcal{A}$ controls all other users and holds their private and public keys. $\mathcal{A}$ may obtain the ephemeral keys used by A and B except for the session attacked. The attack consists of the following phases:

Phase 1: a number of times the protocol is executed between A and B as well as between A or B and the participants controlled by $\mathcal{A}$. For each of these interactions $\mathcal{A}$ may demand revealing the ephemeral values.

Phase 2: A and B execute the protocol. $\mathcal{A}$ can manipulate any message transmitted, but cannot ask for ephemeral values.

Phase 3: If neither A nor B enters an accepting state, then $\mathcal{A}$ looses. If A (respectively, B), terminates in an accepting state, it chooses a bit b at random. Then $\mathcal{A}$ obtains either the session key K_s kept by A (if $b = 0$), or a random key R (if $b = 1$).

Phase 4: it is executed exactly as Phase 1.

Finally, $\mathcal{A}$ answers $\overline{b}$ and wins, if $b = \overline{b}$.

Note that inability to distinguish between the session key and a random key witnesses that no substantial property of the session key can be deduced by $\mathcal{A}$.

5.1 Security Proof

We gradually simplify the attack scenario without substantial changes of adversary's advantage. The initial attack game is described in Sect. 5. The core property of authentication is presented by the following lemma:

Lemma 1. *Assume that CDH Problem is hard. Let y be a element such that discrete logarithm of y with respect to g is unknown. Let c be chosen at random. Then it is infeasible to provide an element c' and (K, a) such that K is a solution for CDH Problem for c and c' and simultaneously $c' = y^a$.*

Proof. Assume conversely that it is possible to present such (K, a). Then we show that it would be able to solve CDH Problem. Given an instance (u, v) of CDH choose r at random and set $y := v^r$. Then choose r' at random and set $c := u^{r'}$. In this way we derive a random instance of the problem concerned in Lemma 1. According to the current assumption derive c' and (K, a). So $K = \mathrm{CDH}(u^{r'}, c')$, where $\mathrm{CDH}(\alpha, \beta)$ stands for the solution of CDH Problem for α and β. However, $c' = y^a$ so $K = \mathrm{CDH}(u^{r'}, y^a) = \mathrm{CDH}(u^{r'}, v^{ra}) = \mathrm{CDH}(u, v)^{r'ra}$. Since we know r, r' and a, we can get $\mathrm{CDH}(u, v)$. □

Corollary 1. *Under the same assumptions as in Lemma 1 it is infeasible to create $E_{K_A}(a, cert_A)$ where $K_A{=}H(K|3)$, $K{=}CDH(c, c')$, and $c'{=}y_A^{H(a)}$.*

Proof. According to Ideal Cipher Model, creating the correct ciphertext is possible only if K_A and a are given. According to the correlated-input secure hash, deriving K_A with a non-negligible probability requires using K. So, getting $E_{K_A}(a, cert_A)$ yields (K, a), which is infeasible by Lemma 1. □

Reducing Phases 1 and 4. One can eliminate all correct interactions between either A or B and a participant controlled by $\mathcal{A}$ from Phases 1 and 4. Indeed, according to Proposition 1 $\mathcal{A}$ can generate transcripts of these interactions with exactly the same probability distribution. The next step is to reveal to the adversary $g^{x_A x_B}$ as it can only increase the advantage of $\mathcal{A}$. However, by Proposition 3 this enables to generate transcripts of correct interactions between A and B with exactly the same probability. Thereby, during Phases 1 and 4 only interactions corrupted by the adversary are left.

Now, let us consider an interaction between B (or A) and D (run by $\mathcal{A}$) in Phase 1 or 4, and initiated by D. As D deviates from the protocol, authentication of D fails and B (or A) sends no second message. So the only message sent by the honest party is the random element c_B, and this can be easily simulated.

The case of an interaction initiated by an honest user, say A, with D controlled by $\mathcal{A}$, is more complicated. There are two subcases: the first is that D can solve CDH Problem for c_A and c_D. This case can be perfectly simulated by $\mathcal{A}$: it chooses a, and proceeds as described by the protocol apart from derivation of K which is done according to the subcase assumption. In the other case, the adversary becomes the ciphertext $E_1 = Enc_{K_A}(a, cert_A)$. However, since $\mathcal{A}$ cannot derive K, we can replace K_A by a random key using correlated-input secure hash assumption. Then, according to the Ideal Cipher Model we can replace the ciphertext E_1 with a random string of the same length.

Attacking Interactions between A and B. The only interactions in Phases 1 and 4 that are left are interactions between A and B corrupted by $\mathcal{A}$. As $\mathcal{A}$ controls the communication channel, we may assume that the following messages are exchanged (the elements with an overline come from $\mathcal{A}$):

- between A and $\mathcal{A}$: c_A, $\overline{c_B}$, E_1, $\overline{E_2}$
- between $\mathcal{A}$ and B: $\overline{c_A}$, c_B, $\overline{E_1}$, E_2

We consider a number of cases depending on the behavior of $\mathcal{A}$.

Case 1: $\mathcal{A}$ cannot derive $\mathrm{CDH}(c_A, \overline{c_B})$.
In this case $\mathcal{A}$ gets a ciphertext E_1 obtained with an unknown key K_A. According to the Ideal Cipher Model assumption, $\mathcal{A}$ cannot get any information about the plaintext or transform it a controlled way. So essentially the adversary may either use $\overline{E_1} = E_1$ or to ignore E_1 when constructing $\overline{E_1}$. In the first case B will accept it provided that $c_A = \overline{c_A}$ and $\mathrm{CDH}(c_A, \overline{c_B}) = \mathrm{CDH}(\overline{c_A}, c_B)$, that is when $c_A = \overline{c_A}$ and $c_B = \overline{c_B}$. In the second case B will not accept $\overline{E_1}$ with a high probability. So we have two cases:

- up to the third step, the execution of the protocol is not disturbed by the adversary,
- there are some modifications by the adversary, but B rejects after getting $\overline{E_1}$ and the ciphertext E_1 can be replaced by a random string.

Consequently, performing the last step (delivery of $\overline{E_2}$) can be done either according to the protocol or simulated by $\mathcal{A}$ (as B is silent).

If the whole protocol is executed without modifications of $\mathcal{A}$, then it can be eliminated from Phases 1 and 4, as already observed. So in all cases we can eliminate such interactions from Phase 1 and 4.

Case 2: $\mathcal{A}$ can derive $\mathrm{CDH}(c_A, \overline{c_B})$.
It means in particular that $c_B \neq \overline{c_B}$. In this case the answer E_1 from A can be simulated by $\mathcal{A}$, as c_A can be generated by an oracle as y_A^a. Consequently, by Lemma 1 the adversary $\mathcal{A}$ cannot create $\overline{E_1}$ that is accepted by B.

Nevertheless, $\mathcal{A}$ can continue interacting with A. However, in this case providing $\overline{E_2}$ and accepting it by A occurs with a negligible probability only. Indeed, the only input from B is a random element c_B which can be simulated.

We conclude that it is possible to simulate the interaction in this case and that neither B nor A enters an accepting state.

Phase 2. We are left with a game consisting of Phases 2 and 3. First we consider the case that B enters an accepting state. This means that E_1 corresponds to c_A received by B and c_B sent by B. According to the Ideal Cipher Model this may occur with a non-negligible probability only if E_1 has been created with the key K_A as computed by B. Indeed, the plaintext contains $cert_A$, which is fixed, so a different key for the same ciphertexts would lead to a plaintext not containing $cert_A$. (Also a can be checked against y_A and c_A.)

The presence of the correct a witnesses that E_1 originates from a party that used the same c_A as received by B. On the other hand, to get K_A it is necessary to use K, apart from a negligible probability. In turn, deriving $K = c_B^{x_A h_A}$ for known (but random) c_B, known h_A, and y_A, but without x_A is equivalent to solving CDH Problem for c_B and y_A. As we assume that the DDH Problem is hard, this is infeasible. So B can assume that E_1 have been created by a party holding the key x_A, that is by A. It means that c_A originates from A.

Now, let us argue why adversary $\mathcal{A}$ cannot distinguish between the right session key and a random key. Note that all messages sent by A and B correspond to a correct protocol execution (maybe the last message from B to A is not delivered correctly). Then we may reveal the values of a, b, K_A, K_B, and refer to correlated-input secure hash function condition, where the random parameters used by the circuits are x_A, x_B.

The same argument can be applied to cover the case that A enters an accepting state.

6 Leaking Ephemeral Keys

Ephemeral values may be implemented in a less secure way than long-time secret keys. Therefore it is necessary to consider consequences of revealing them. In particular, the attack may be performed against A that interacts with B which is controlled by an adversary. We draft here two cases:

Attempt to Learn x_A or x_B. We concern the extreme case that the adversary is getting a, b as well as the private key x_B and attempts to learn x_A. However, in this case the messages exchanged between A and B can be perfectly simulated according to Sect. 4. So any attack executed in this way can be performed off-line with the same effect. In turn, the off-line attack can be used as an attack against the Discrete Logarithm Problem: we choose at random the values a, b, x_B, derive a protocol description and run the off-line adversary on these data.

Attempt to Learn a Session Key. Assume that we are given a transcript of an interaction consisting of c_A, c_B, E_1, E_2 and a and b used for this interaction. Ability to learn anything on the session key is described by the following game:

Game 0.

```
    choose $a$, $b$ at random, $h_A := H(a|0)$, $h_B := H(b|0)$, $c_A := y_A^{h_A}$, $c_B := y_B^{h_B}$
    $K := g^{x_A x_B h_B h_A}$,   $K_A := H(K|1)$, $K_B := H(K|2)$
    $E_1 := \mathit{Enc}_{K_A}(a, \mathit{cert}_A)$, $E_2 := \mathit{Enc}_{K_B}(b, \mathit{cert}_B)$
    choose $u$ at random
    if $u = 0$, then $R := H(K|3)$, otherwise choose $R$ at random
    $u' := \mathcal{A}(a, b, E_1, E_2, R, y_A, y_B)$
```

The adversary wins, if $u' = u$.

Below we consider a modified version of this game, where E_1 and E_2 are generated in a different way.

Game 1.

```
    choose $a$, $b$ at random, $h_A := H(a|0)$, $h_B := H(b|0)$, $c_A := y_A^{h_A}$, $c_B := y_B^{h_B}$
    $K := g^{x_A x_B h_B h_A}$,   $K_A := H(K|1)$, $K_B := H(K|2)$
    choose $E_1$ and $E_2$ at random
    choose $u$ at random
    if $u = 0$, then $R := H(K|3)$, otherwise choose $R$ at random
    $u' := \mathcal{A}(a, b, E_1, E_2, R, y_A, y_B)$
```

A difference between Game 0 and Game 1 may be observed only if $\mathcal{A}$ asks the encryption oracle $\mathcal{O}$ a query containing K_A or K_B. Then decrypting E_1 or E_2 may yield wrong results (in the Game 1, $\mathcal{O}$ does not know a and b, so with a high probability it will choose the plaintext inconsistently). However, if $\mathcal{A}$ may generate K_A or K_B

with a non-negligible probability, then we can construct a distinguisher for the DDH Problem, just as in the proof of Proposition 2.

Now let us clean up by eliminating parameters unused by the adversary or random. Thereby we get the following game:

Game 2.

```
choose $a$, $b$ at random, $h_A := H(a|0)$, $h_B := H(b|0)$, $K := g^{x_A x_B h_B h_A}$
choose $u$ at random
if $u = 0$, then $R := H(K|3)$, otherwise choose $R$ at random
$u' := \mathcal{A}(a, b, R, y_A, y_B)$
```

Now, it is easy to see that Game 2 could be directly used for solving the DDH Problem: given a candidate triple (U, V, Z), we choose a, b, r_1, r_2 at random, put $y_A := U^{r_1}$, $y_B := V^{r_2}$ and $R := H(Z^{r_1 r_2 h_A h_B}|3)$. Then we give a, b, R, y_A, y_B to $\mathcal{A}$. (Note that r_1, r_2 are used to randomize the input.)

Note that if Z is random, then R created as above is not a random value, but a hash value of a random value. However, any difference in behavior of $\mathcal{A}$ in case of random R and $R := H(Z^{r_1 r_2 h_A h_B}|3)$ for a random Z would lead to a procedure that distinguishes the values of the form $H(S|3)$ from the random strings of the same length. For correlated-input secure hash functions this is impossible.

The above argument can be extended to the case when we have a number of interactions between A and B and the corresponding ephemeral keys. In this case we formulate the following game for k interactions:

Game 0'.

```
choose $a_i$, $b_i$ at random, $h_{i,A} := H(a_i|0)$, $h_{i,B} := H(b_i|0)$, for $i \leq k$,
$c_{i,A} := y_A^{h_{i,A}}$, $c_{i,B} := y_B^{h_{i,B}}$, for $i \leq k$,
$K_i := g^{x_A x_B h_{i,B} h_{i,A}}$, for $i \leq k$,
$K_{i,A} := H(K_i|1)$, $K_{i,B} := H(K_i|2)$, for $i \leq k$,
$E_{i,1} := \mathit{Enc}_{K_{i,A}}(a_i, \mathit{cert}_A)$, $E_{i,2} := \mathit{Enc}_{K_{i,B}}(b_i, \mathit{cert}_B)$, for $i \leq k$,
choose $u$ at random, choose $S$ at random
if $u = 0$, then $R_i := H(K_i|3)$, otherwise $R_i := H(S^{h_{i,B} h_{i,A}}|3)$, for $i \leq k$
$u' := \mathcal{A}(a_1, \ldots, a_k, b_1, \ldots, b_k, E_{1,1} \ldots, E_{k,1}, E_{1,2} \ldots, E_{k,2}, R_1, \ldots, R_k)$
```

After making essentially the same transformations we get a proof for the following theorem:

Theorem 2. *Assume that H is a correlated-input secure hash function and that the DDH Problem is hard. Then, in the Ideal Cipher Model it is infeasible to derive any information on the session keys of MRI given the messages exchanged and the ephemeral keys a, b used for these interactions.*

7 Forward Security

Another problem we have to concern is that at some moment the private key x_A is disclosed. This may occur due to physical attack with techniques unknown at the time of the system deployment. In this scenario the adversary has no access to the ephemeral keys – as they should be stored in a volatile memory or erased after usage. So the attack scenario can be described by the following game:

Game 0.

```
choose $a$, $b$ at random, $h_A := H(a|0)$, $h_B := H(b|0)$
$c_A := y_A^{h_A}$, $c_B := y_B^{h_B}$
$K := g^{x_A x_B h_B h_A}$, $K_A := H(K|1)$, $K_B := H(K|2)$
$E_1 := Enc_{K_A}(a, cert_A)$, $E_2 := Enc_{K_B}(b, cert_B)$
choose $u$ at random
if $u = 0$, then $R := H(K|3)$, otherwise choose $R$ at random
$u' := \mathcal{A}(x_A, x_B, E_1, E_2, R)$
```

The adversary wins if $u' = u$. Following almost exactly the same argument as in the proof of Theorem 2 we get the following result (in fact, the results holds also under assumption of semantic security):

Theorem 3. *Assume that H is a correlated-input secure hash function and that Decisional DDH Problem is hard. Then in the Ideal Cipher Model it is infeasible to derive any information on the session key of MRI executed between A and B, given the messages exchanged and the private keys x_A, x_B.*

7.1 Malicious Implementations

If a protocol is implemented in a black box device, then a user is exposed to malicious implementations that behave like the original protocol – no procedure based on the regular output may detect any difference – but a party holding appropriate secret (not stored in the device) gets access to private data of the user (see *kleptographic attacks*, e.g. [19]). The key mechanism of kleptographic attacks is to use a pseudorandom parameter that can be derived by the device (from its internal values) and the attacker (from the previous output of the device and the secret of the attacker). As an authentication protocol cannot be deterministic it seems that there is always room for such an attack.

Let us discuss shortly susceptibility of the MRI protocol to such attacks. As the long time secrets x_A are used for exponentiations only, x_A can be implemented in ROM with no access to other operations. In particular, for ROM it is impossible to manipulate the code. The code for the remaining parts of MRI may be included e.g. in a smart card applet, where manipulations are much easier. Nevertheless, at worst the applet may serve as an oracle for computing values d^{x_A}, where the numbers d are given. This may slightly ease a cryptanalytic attack against x_A, but not expose x_A directly.

The other target of the adversary is to derive a session key. Note that leaking the ephemeral value a (or h_A) without x_A does not enable to derive a session key: given c_B and h_A we still need x_A to obtain $K = c_B^{h_A x_A}$. So the leakage must be more complicated than just based on malicious way of computing h_A.

Finally, we have to be aware that MRI, like any other protocol with pseudorandom values, enables a limited hidden channel. Simply, in order to leak a short bit string $\kappa = k_0 k_1 \ldots k_m$ we leak a few bits in each c_A. Namely, the malicious implementation chooses a until $H(Y^{h_A})$ has $k_0 \ldots k_m$ as leading bits. If $Y = g^z$ and z is held by the adversary, then κ can be recomputed from $H(c_A^z)$. On average, 2^m trials are necessary, so m cannot be large, especially for smart cards.

References

1. BSI: Advanced Security Mechanisms for Machine Readable Travel Documents 2.1, parts 1-3. Technische Richtlinie TR-03110-1 (2012)
2. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
3. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. Cryptology ePrint Archive, Report 2005/176 (2005)
4. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28(2), 119–134 (2003)
5. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
6. Lee, J., Park, J.H.: Authenticated key exchange secure under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/344 (2008)
7. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Cryptology ePrint Archive, Report 2007/123 (2007)
8. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
9. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just fast keying: Key agreement in a hostile internet. ACM Trans. Inf. Syst. Secur. 7(2), 242–273 (2004)
10. Cheng, Z., Chen, L., Comley, R., Tang, Q.: Identity-based key agreement with unilateral identity privacy using pairings. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 202–213. Springer, Heidelberg (2006)
11. Chien, H.-Y.: ID-based key agreement with anonymity for ad hoc networks. In: Kuo, T.-W., Sha, E., Guo, M., Yang, L.T., Shao, Z. (eds.) EUC 2007. LNCS, vol. 4808, pp. 333–345. Springer, Heidelberg (2007)
12. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE-protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
13. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: The PACE|AA protocol for machine readable travel documents, and its security. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 344–358. Springer, Heidelberg (2012)
14. Raimondo, M.D., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 400–409. ACM (2006)
15. Hanzlik, L., Kluczniak, K., Kubiak, P., Kutyłowski, M.: Restricted identification without group keys. In: Min, G., Wu, Y., Liu, L.C., Jin, X., Jarvis, S.A., Al-Dubai, A.Y. (eds.) TrustCom, pp. 1194–1199. IEEE Computer Society (2012)
16. Goyal, V., O'Neill, A., Rao, V.: Correlated-input secure hash functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer, Heidelberg (2011)
17. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: Domain-specific pseudonymous signatures for the German identity card. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 104–119. Springer, Heidelberg (2012)
18. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
19. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)

Trust Views for the Web PKI

Johannes Braun, Florian Volk, Johannes Buchmann, and Max Mühlhäuser

Technische Universität Darmstadt/CASED
Hochschulstraße 10, 64283 Darmstadt, Germany
{jbraun,buchmann}@cdc.informatik.tu-darmstadt.de,
florian.volk@cased.de, max@informatik.tu-darmstadt.de

Abstract. The steadily growing number of certification authorities (CAs) assigned to the Web Public Key Infrastructure (Web PKI) and trusted by current browsers imposes severe security issues. Apart from being impossible for relying entities to assess whom they actually trust, the current binary trust model implemented with the Web PKI makes each CA a single point of failure. In this paper, we present the concept of *trust views* to manage variable trust levels for exactly those CAs actually required by a relying entity. This reduces the set of trusted CAs and minimizes the risk to rely on malicious certificates issued due to CA failures or compromises.

1 Introduction

The Web PKI is one of the largest and most important cryptographic systems. The core of the Web PKI is the ecosystem of CAs that are responsible for the issuance and the maintenance of SSL certificates. These certificates are issued to web service providers and are used in the SSL/TLS protocols. Thus, the Web PKI enables authentication of web servers and subsequently the establishment of secure connections between web browsers and services like e-banking or e-commerce, where privacy, confidentiality, and integrity are often indispensable.

However, the Web PKI fails in many points to provide the desired security [7,9, 10]. One serious problem is that the Web PKI does not scale with the enormous size of the Internet. For the sake of interoperability (i.e., as much legitimate web service certificates as possible should be verifiable) the number of CAs, which are fully trusted by default in current browsers and operating systems, has continuously been growing over the past. Currently, there are approximately 1.500 trusted CAs [6]. As each of these trusted CAs can sign certificates for any web service or domain, trusting a single malicious CA, i.e., one that is in fact not trustworthy, can break the whole Web PKI's security. An adversary, who is in possession of a fake certificate that was issued by one of the trusted CAs, can potentially intercept the complete communication between any Internet user and the certified web server without the user even noticing the attack. Thus, with each additional CA, the risk of trusting a malicious or defective CA increases. Several security incidents in the last time clearly show that this is more than just a hypothetical threat [5, 9, 11, 12].

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 134–151, 2014.

Blacklisting CAs or revoking malicious certificates are the reactions to security incidents. However, as these mechanisms are reactive, they have an inherent delay exposing the users at risk until the detection of the threat. As explained, the risk grows proportional to the number of CAs a user trusts.

In this paper, we propose a new approach to reduce this risk by reducing the number of trusted CAs to those that are really required by the users. Recent experiments with browser histories of different users have shown that on average, a user only depends on a small subset of CAs. The size of which lies in the range of 10% of the CAs available and trusted by default in the Web PKI [1]. Additionally, among the trusted CAs, we introduce variable trust levels to enable more fine grained trust decisions. According to the value-at-stake, a trust level might be sufficient or not to consider a connection to a web service as secure.

To achieve this, we present the concept of *trust views* that serve as a local and user dependent knowledge base for trust decisions. We present the mechanisms for the establishment and the management of the trust view. We implement learning processes and define decision rules by employing computational trust models. The real trustworthiness of CAs is approximated by a subjective probabilistic trust value. Our approach allows an adaptation to the requirements of the user and automated trust decisions based on defined decision rules. Our system focuses on applicability, thus it only uses data which is already available or is collected over time. However, our system is open to be extended with additional information sources.

The paper is organized as follows. Section 2 describes the Web PKI and our security model. In Section 3 we introduce computational trust and present related work. Afterward, in Section 4 we present the trust view concept. We describe challenges and how a trust view is modeled. Then we describe the initialization mechanisms for trust views and give the relevant algorithms for trust validation and the update of the trust views. In Section 5 we evaluate our approach and discuss limitations. We end with a conclusion and future work in Section 6.

2 Web PKI and Security Model

2.1 The Web PKI

Secure Internet connections between web browsers and web servers in general rely on public key cryptography to authenticate web servers and establish session keys. Public key cryptography requires the knowledge of key pairs: a private key that is only known to the owner of the key pair (in our case a web server) as well as a public key, which must be known to everyone who wants to establish a secure connection to the owner of the associated private key. A public key is bound to an identity via a digital *certificate* according to the X.509 standard [4]. Whenever a relying entity contacts a web server and successfully establishes an SSL/TLS connection using the public key in the web server's certificate, the relying entity can be sure that the web server knows the private key matching the public one. As the certificate binds the public key to an identity, the relying entity can be sure about the authenticity of the web server.

As it is impossible to exchange certificates directly between all web servers and all browser users (the relying entities), the Web PKI uses a hierarchical but tightly interwoven structure of CAs that digitally sign certificates. If a certificate is signed by a trusted CA, the authenticity of a web server that employs the certificate is transitively trusted. The Web PKI has a set of *Root CAs*. Their public keys are usually distributed within trusted lists called *root stores*, along with operating systems and browsers. The Root CAs act as basis for the whole PKI. Root CAs sign certificates for *subordinate CAs* (Sub CAs) which themselves sign certificates for other Sub CAs and web servers. This way, a hierarchical structure is created. The chain of certificates starting with a Root CA's certificate and ending with a web server's certificate is called *certification path*. The process of checking the certification path for correctness and validity is called *path validation* [4].

For a relying entity, in order to be convinced of the *key legitimacy* of a public key k, namely to be convinced whether a public key k in a certificate belongs to the identity contained in the subject field of the certificate, two things are required [20,23,33]. First, the relying entity must be convinced of the key legitimacy of the CA's key that was used to sign the certificate. Second, the relying entity must trust the CA to issue trustworthy certificates which is called *issuer trust* in the CA.

In the Web PKI, issuer trust and key legitimacy are binary. Any certificate signature that can be verified using the root store is absolutely trusted. Although CAs achieve different qualities of service, this is currently not reflected within trust decisions. Different CAs implement different schemes to verify identities of the key owners they sign certificates for and employ different security mechanism. But, for example, a certificate containing a superficially verified identity appears to be as trustworthy as a certificate where the contained identity was checked thoroughly.

2.2 Security Model

In the model exist two entities e_1 and e_2. e_1 establishes an SSL/TLS connection to e_2. The problem is to decide if the connection is trustworthy for e_1.

A connection is trustworthy for e_1 if the public key k of e_2 that was used in the SSL/TLS connection establishment is trusted by e_1 to be a valid public key of e_2. This requires:

1. e_1 has a valid certificate C that binds k to e_2.
2. e_1 trusts in the issuer of C.

Requirement 1 is a standard PKI issue. To fulfill requirement 1, e_1 needs to have a certification path $p = (C_1, ..., C_n)$ such that

1. C_n=C
2. p passes path validation

Requirement 2 is fulfilled if p additionally passes *trust validation*. Explicit trust validation is not incorporated in the current deployment of the Web PKI. We show how this can be realized with the concept of *trust views* and explain how this enables to reduce the number of actually trusted CAs and therewith the risk of relying on maliciously issued certificates. We first introduce computational trust and present related work.

3 Computational Trust and Related Work

3.1 Computational Trust

Computational trust is a means to support users in making decisions under uncertainty, e.g., under incomplete information. Jøsang defines *decision trust* in [19]:

> *Decision trust is the extent to which a given party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.*

Starting from recommendations, experiences from previous interactions, and context-related indicators of trustworthiness, computational trust models calculate an approximation for the quality of future interactions. For this paper, the CertainTrust trust model by Ries [26] is used. CertainTrust was extended with CertainLogic, a set of operators to combine CertainTrust opinions. These operators are similar to those of propositional logic, but consider the inherent uncertainty of CertainTrust opinions.

CertainTrust can handle two ways of expressing trust-related information:

- The *experience space* collects results from interactions as binary experiences, i.e., an interaction was either positive or negative.
- The *opinion space* uses a triple (t, c, f) to express an opinion o_S about a statement S. The value $t \in [0; 1]$ represents the trust in the correctness of the statement, while the certainty $c \in [0; 1]$ represents the probability that t is a correct approximation. c scales with the amount of information (for example, the number of collected experiences): the more information available, the more reliable is the approximation. Finally, $f \in [0; 1]$ defines a context-specific, initial trust value in case no information was collected, yet. This parameter serves as a baseline and represents *systemic trust*.

There exists an ambilateral mapping between the experience space and the opinion space by parametrizing a Bayesian probability density function with the amount of positive and negative experiences. For details, see [25]. In this paper, trust information is collected in the experience space but the opinion space is used to combine trust statements about different CAs. Opinions can be updated with newly collected positive or negative experiences by mapping the opinion into the experience space, adding the new experience to either the number of positive or negative experiences and mapping those back to the opinion space.

There are several operators to combine different opinions. From two opinions about two independent statements a combined opinion about the statement regarding the truth of both input statements is computed with the AND-Operator of CertainLogic [26]:

Definition 1 (CertainLogic AND-Operator). *Let A and B be independent statements and the opinions about these statements be given as $o_A = (t_A, c_A, f_A)$ and $o_B = (t_B, c_B, f_B)$. Then, the combined opinion on the statement regarding both A and B is defined as follows:*

$$\begin{aligned}
o_A \wedge o_B &= (t_A \wedge t_B, c_A \wedge c_B, f_A \wedge f_B) \text{ with} \\
c_{A\wedge B} &= c_A + c_B - c_A c_B - \\
&\quad \frac{(1-c_A)\, c_B\, (1-f_A)\, t_B + c_A\, (1-c_B)\, (1-f_B)\, t_A}{1 - f_A f_B} \\
\text{if } c_{A\wedge B} = 0 \text{: } t_{A\wedge B} &= 0.5 \\
\text{if } c_{A\wedge B} \neq 0 \text{: } t_{A\wedge B} &= \frac{1}{c_{A\wedge B}} (c_A c_B t_A t_B + \\
&\quad \frac{c_A(1-c_B)(1-f_A) f_B t_A + (1-c_A) c_B f_A (1-f_B) t_B}{1 - f_A f_B}) \\
f_{A\wedge B} &= f_A f_B
\end{aligned}$$

The CertainLogic AND-Operator is commutative.

From opinions, an expectation can be computed. It represents the expectation for future behavior. In CertainTrust, the expectation of an opinion o_A is defined as

$$E(o_A) = t_A \cdot c_A + f_A(1 - c_A)$$

Herein, with increasing certainty (which means that a larger amount of experiences is available), the influence of the initial trust f ceases.

3.2 Related Work

The multitude of problems and disadvantages of the currently deployed Web PKI is described by well known researchers [7,9,10]. Monitoring of the Web PKI reveals its enormous size and shows that indeed malpractices are common [6,14].

Many attempts exist to circumvent the problems imposed by possible CA failures and thus to enhance Internet security. Certificate pinning (e.g., [8, 24]) means that relying entities store certificates of formerly accessed websites. Based on the *trust on first use approach*, it implies that a possible adversary must be present during the first connection establishment to the website. Unfortunately, this either implies that each CA is trusted equally in case a new web page is accessed or that the trust decision is transferred to the relying entity requiring it to have PKI expertise. Also, the approach suffers from the problem how and when to allow pinned certificates to be exchanged (e.g. due to certificate expiry). Notarial solutions [2,16,21] maintain databases containing formerly observed certificates and can be queried to reconfirm the authenticity of a specific certificate, sometimes also involving consensus decisions of several independent notary servers. In this approach, trust is deferred from the CAs to a majority of notaries. Central instances come with availability and scalability problems. Also, privacy protection issues and delays due to the communication overhead are clear disadvantages.

The enhancement of PKI with trust computation has been proposed by many researchers. The CertainTrust model and CertainLogic [26] used by us are equivalent to the Beta Reputation System and Subjective Logic, both by Jøsang et al. [17, 18], as these models both rely on binary experiences that are combined using a Bayesian approach with beta probability density functions. A survey on different trust models that rely on this computational approach and similar ones can be found in the surveys by Jøsang et al. [19] and Ruohomaa et al. [27]. Jøsang proposes an algebra for trust assessment in certification chains in [20] but mainly addresses trust networks similar to PGP [33]. Huang and Nicol [15] also define another trust model for trust assessment in PKI. Both approaches require trust values recommended by the intermediates to evaluate trust chains. Such recommendations are in general not included within commercial certificates and we do not expect any entity to pay for a certificate that includes a low trust value. Different certificate classes like domain validated (DV) or extended validation (EV) can be indicators for such trust values, but in our opinion these are not sufficient for trust evaluation.

Other researchers base trust evaluation in CAs on their policies and the adherence to those [3, 29]. This requires policy formalization [3, 29, 31] for automated processing. Such formalized policies are not provided by the CAs, and are in general far to complex to be evaluated by the relying entities. Therefore, such approaches require technical and legal experts to process policies [30].

Our solution builds on the techniques of previous works, however there are several fundamental differences. We combine different mechanisms and use them as building blocks to solve separate subproblems. The novelty thereby is to locally and user-specific limit the number of trusted CAs to those the user actually requires. Different from pure pinning and notarial solutions the CAs in this user-specific set have different trust levels and might even be fully trusted depending on the context. Thus our solution does not require an additional check of each (new) certificate and provides a trade-off between overhead and solely relying on CAs. While we make use of established computation models for trust evaluation, we base it on local experiences of the user instead of using recommendations of trust values within certificates or the evaluation of certificate policies and expert opinions. Thus, our system can work autonomously and only requires notarial reconfirmation, when not enough local experience has been collected so far. Furthermore, the management of local experiences guarantees, that independent from the CA's global reputation it is not trusted without an additional check, if the CA has never been observed by the user before. This aims at also protecting the user from malfunctions of CAs that in general follow good security practices but are actually irrelevant for the user.

4 Trust View and Trust Validation

The purpose of establishing a trust view is to enable explicit trust validation thereby locally reducing the number of trusted CAs on a per-user level. The differences in the trust needed for different applications are considered during trust validation. For example, there is a difference in the trust needed to visit a search engine and the trust needed to supply an online-shopping web site with your credit card information.

4.1 Challenges

The set of CAs required by a user is not fixed but changes over time. The challenge herein is to establish and manage a trust view in a dynamic way. We identified the following constraints for dynamically updating the set of trusted CAs as well as assigning trust levels to them:

1. **Minimal user involvement**: an informed assessment of the quality of a CA's certification processes is beyond the capabilities of the average Internet user [13, 28].
2. **Incomplete information on CA processes**: data on the quality of a CA's certification process might be incomprehensible or not available at all.
3. **Incomplete information on user requirements**: in general, the web services that a user will contact in the future are unknown and therefore also the required CAs to verify the certificates of such web services.

4.2 The Trust View

For trust validation, entity e_1 has a *trust view* View. The trust view is the local knowledge base of e_1 and contains all previously collected information about other entities and their keys. It is built incrementally during its use for trust validation. The trust view of e_1 consists of:

- a set of trusted certificates
- a set of untrusted certificates
- a set of public key trust assessments

The trusted certificates are all certificates that have previously been used to establish a trustworthy connection to another entity. The untrusted certificates are those certificates, for which the connection was evaluated untrustworthy. Furthermore, there is one public key trust assessments for each pair of (*public key*, *CA name*) that was contained in a previously evaluated certification path. A trust assessment represents all information collected for the respective pair during prior trust validations.

A public key trust assessment TA is a tuple $(k, ca, S, o_{kl}, o_{it})$, where

- k is a public key.
- ca is the name of a certification authority.

- S is a set of certificates for k. The subject of these certificates is ca. This set contains all the certificates with subject ca and public key k that have previously been verified by e_1.
- o_{kl} is an opinion. It represents the opinion of e_1 whether k belongs to ca (key legitimacy of k).
- o_{it} is an opinion. It represents the trust of e_1 in ca to issue trustworthy certificates (issuer trust in ca, when using k).

In order to decide whether the connection to entity e_2 is trustworthy, entity e_1 runs the trust validation algorithm (cf. Section 4.4).

4.3 Initialization of Trust Assessments

A trust assessment $TA = (k, ca, S, o_{kl}, o_{it})$ is initialized whenever a pair $(\textit{public key}, \textit{CA name})$, for which there is no trust assessment in the trust view View, is observed within a CA certificate C. We assume that a root store is available during initialization. Then, TA is initialized as follows:

- $k = \textit{public key}$
- $ca = \textit{CA name}$
- $S = \{C\}$
- $o_{kl} = (1, 1, 1)$ if the CA is a Root CA, else $o_{kl} = \textit{unknown}$.
- $o_{it} = (0.5, 0, 0.5)$ if no prior information is available about the issuer trust of the CA. Else if for $1 \leq i \leq n$ there are trust assessments $TA_i \in \mathsf{View}$ with $C_i \in S_i$, where the issuing CA of C_i is equal to the issuing CA of C, then $f = (\sum_{i=1}^{n} E(o_{it,i}))/n$ and $o_{it} = (0.5, 0, f)$.

The key legitimacy is set to complete ($o_{kl} = (1, 1, 1)$) for Root CA keys as these keys are confirmed via the root store. For other CA keys, key legitimacy is computed during trust validation as long as key legitimacy is *unknown*. During the evolution of the trust view, key legitimacy may be changed to complete as soon as enough evidence has been collected. We discuss this in Section 4.5.

The issuer trust $o_{it} = (0.5, 0, 0.5)$ reflects that no experiences have been collected and that the CA may either be trustworthy or not. If the new CA is certified by a CA that certified several other CAs, for which experiences have already been collected, we use the average over the expectations of the respective issuer trusts for initialization. The reason is that a CA evaluates a Sub CA before signing its key, and thus these Sub CAs are assumed to achieve a similar level of issuer trust, like a stereotype.

Optimally, further information is collected for initialization. Our system is open for such extensions. Further information can be gathered from policy evaluation as, e.g., proposed by Wazan et al. [29,30]. A drawback of this approach is its need for some kind of expert or expert system to evaluate the certificate policies and practice statements, because these documents cannot be processed automatically at the time being. So far, no such services are available in practice. Yet, given such additional data, it can be mapped into an opinion and integrated into the initialization process.

Bootstrapping
Despite the fact that an entity will often access the same services and see the same CAs repeatedly [1], it takes a certain time until enough experiences are collected such that the system may operate autonomously. Therefore, a bootstrapping procedure is required to face possible delays and usability problems due to the involvement of additional validation services (cf. Section 4.4 for details). A possibility for such a bootstrapping procedure is to scan the browsing history. From the history, the services that are accessed via *https* can be identified and the respective certification paths can be downloaded. The paths can then be used to bootstrap the trust view. This initial bootstrapping is only to be performed once and afterward, the system can mainly fall back on the collected experiences.

4.4 Trust Validation

We now describe the trust validation algorithm. It takes a trust view of entity e_1 and a certification path for the certificate of entity e_2 as input and computes the key legitimacy of e_2's key to decide whether a connection established with e_2's key is to be considered trustworthy. The decision depends on the security criticality of the application that is to be secured by the connection from e_1 to e_2. The information available in the trust view may not be sufficient to complete the trust validation. In such a case, validation services are used as a fall back mechanism. We present the detailed algorithm in the following:

Input:

- The certification path $p = (C_1, ..., C_n)$
- The trust view View of e_1
- A security level $l \in [0; 1]$ for C_n. l is selected by e_1 and represents the security criticality of the application that is to be secured by the connection from e_1 to e_2.
- A required certainty $rc \in [0; 1]$. rc is selected by e_1 and represents on how much previous information the decision must be based to be accepted.
- A list of validation services $VS = (vs_1, ..., vs_j)$ with outputs $R_i = vs_i(C) \in \{trusted, untrusted, unknown\}, 1 \leq i \leq j$ on input of a certificate C.

Output: $R \in \{trusted, untrusted, unknown\}$

The algorithm proceeds as follows:

1. If C_n is a trusted certificate in View then $R \leftarrow trusted$
2. If p contains a certificate that is an untrusted certificate in View then $R \leftarrow untrusted$
3. If C_n is not a certificate in View then
   (a) For $1 \leq i \leq n - 1$ set k_i to the public key in C_i and ca_i to the subject in C_i.
   (b) Initialize the trust assessments for pairs (k_i, ca_i) for which there is no trust assessment in View (as described in Section 4.3). Store the new trust assessments in the list TL.

```
    (c) For $1 \leq i \leq n-1$ set $o_{kl,i}$ to the key legitimacy of $k_i$ and $o_{it,i}$ to the
        issuer trust assigned to $k_i$ in View.
    (d) Set $h = \{max(i) : o_{kl,i} = (1,1,1)\}$
    (e) Compute $o_{kl,n} = (t,c,f) = o_{it,h} \wedge o_{it,h-1} \wedge \cdots \wedge o_{it,n-1}$
    (f) Compute the expectation $exp = E(o_{kl,n})$
    (g) If $exp \geq l$ then $R \leftarrow trusted$
    (h) If $exp < l$ and $c \geq rc$ then $R \leftarrow untrusted$
    (i) If $exp < l$ and $c < rc$ then
        i. For $1 \leq i \leq j$ query validation service $vs_i$ for $C_n$ and set $R_i = vs_i(C_n)$.
        ii. Set $R_c = cons(R_1, ..., R_j)$ to the consensus on $(R_1, ..., R_j)$, then $R \leftarrow R_c$.
    (j) Update View. (See Section 4.5 for details.)
4. Return $R$
```

According to previous works [20,23,33], the key legitimacy of a key is computed as the key legitimacy of the CA's key in conjunction with the issuer trust in the CA : $o_{kl} = o_{kl,CA} \wedge o_{it,CA}$. The computation of the key legitimacy based on a certification path with length greater than one follows directly from chaining this rule and the fact that the key legitimacy of the first key k_h in the path is $o_{kl,h} = (1,1,1)$. Such a key always exists as this holds at least for Root CA keys. Thus, $o_{kl,n} = (t,c,f) = o_{it,h} \wedge o_{it,h-1} \wedge \cdots \wedge o_{it,n-1}$ as for the CertainLogic AND operator holds if $o_A = (1,1,1)$ then $o_A \wedge o_B = o_B$.

Security Levels

e_1 assigns security levels to classes of applications according to their value-at-stake (cf. [29] for a similar approach). A security level is a real number between 0 and 1. The higher the security level is, the higher is the required key legitimacy for a connection to be evaluated trustworthy. The assignment of security levels is a subjective process and relies on the risk profile of e_1, which is out of scope of this paper. General examples for security levels could be 0.99 for online banking, 0.9 for e-government applications, and 0.6 for social networks. Note, that as the trust validation requires the security level as input, the determination of the class of application is required. While an automated solution, for example, one based on content filtering (as also used to detect phishing sites [32]) or based on analyzing the type of entered data (cf. [22]) is desirable, this is out of scope of this work. In any case, the security level can be indicated by the entity, e.g., by using a visual slide control as part of the browser's user interface.

Validation Services

A certification path containing previously unknown CAs results in a low trust value. On the one hand this is intended, as it leads to the rejection of keys certified by unknown CAs. However, this is not necessarily due to malicious behavior, but due to the lack of information. Thus, whenever the key legitimacy is too low to consider a connection trustworthy, and the certainty is below a threshold rc which is set by e_1, validation services like notary servers (cf. Section 3.2) are queried to reconfirm a certificate. If a certificate is reconfirmed to be authentic, the connection is considered trustworthy. If the validation services reply with *unknown*, i.e., it is unclear if the certificate is trustworthy or not, the algorithm outputs *unknown*. Only in this case, the user is asked for a decision.

As the lack of expertise makes user involvement problematic, the '*unknown*´ case needs to be avoided whenever possible by the use of an adequate set of validation services. Further research on additional information sources and the optimization of the use of validation services is due to future work. Yet, given sufficiently support, involving the user for decision making might be a viable approach, for example, when a bank provides his customers with further information about their certificates.

4.5 Trust View Update

New information needs to be incorporated into the trust view to be available during future trust validations. Based on the output of the trust validation, either positive or negative experiences are collected for the involved trust assessments. Herein, it is important that only strictly new information is collected. Therefore, it is checked if a certificate, which is contained in the considered certification path, is evaluated for the first time based on the state of the trust view. We present the detailed algorithm in the following:

Input:

- A certification path $p = (C_1, ..., C_n)$
- A trust view View
- An output of the trust validation R
- A list of new trust assessments TL
- A list of validation services $VS = (vs_1, ..., vs_j)$ with outputs $R_i = vs_i(C) \in \{trusted, untrusted, unknown\}, 1 \leq i \leq j$ on input of a certificate C.

Output: The updated trust view.

The algorithm proceeds as follows:

```
1. If $R = unknown$ then return $\mathsf{View}$
2. For $1 \leq i \leq n-1$ set $k_i$ to the public key in $C_i$, set $ca_i$ to the subject in $C_i$
   and set $TA_i = (k_i, ca_i, S_i, o_{kl,i}, o_{it,i})$ to the corresponding trust assessments.
3. If $R = trusted$ then
   (a) For $1 \leq i \leq n-1$ do
       i. If $C_i \notin S_i$ add $C_i$ to $S_i$
       ii. If $(i = n-1)$ or $(TA_{i+1} \in TL)$ or $(C_{i+1} \notin S_{i+1})$ then update $o_{it,i}$
           with a positive experience.
       iii. If $TA_i \in TL$ then add $TA_i$ to $\mathsf{View}$.
   (b) Add $C_n$ to $\mathsf{View}$ as trusted certificate.
```

```
4. If $R = untrusted$ then
   (a) Set $h = \{max(i) : TA_i \notin TL$ or the consensus $cons(vs_1(C_i), ..., vs_j(C_i)) = trusted\}$.
   (b) For $1 \leq i \leq h - 1$ do
       i. If $C_i \notin S_i$ add $C_i$ to $S_i$
       ii. If $(TA_{i+1} \in TL)$ or $(C_{i+1} \notin S_{i+1})$ then update $o_{it,i}$ with a positive experience.
       iii. If $TA_i \in TL$ then add $TA_i$ to View
   (c) If $C_h \notin S_h$ add $C_h$ to $S_h$
   (d) If $TA_h \in TL$ then add $TA_h$ to View
   (e) If $C_{h+1}$ is not an untrusted certificate in View then update $o_{it,h}$ with a negative experience.
   (f) Add $C_{h+1}$ to View as untrusted certificate.
5. Return View
```

Example

An exemplary evolution of the trust view is shown in Figure 1. It visualizes the experience collection process. Root CAs are denoted with R-CA, Sub CAs with S-CA. The arrows represent observed certificates.

(a) The system obtains the chain R-CA$_1$ → S-CA$_1$ → EE$_1$. The certificate of EE$_1$ is accepted. A positive experience is added to each involved CA.

(b) The chain R-CA$_1$ → S-CA$_2$ → EE$_2$ is obtained. The certificate of EE$_2$ is accepted. A positive experiences is added to each involved CA.

(c) The chain R-CA$_1$ → S-CA$_2$ → EE$_3$ is obtained. The certificate of EE$_3$ is rejected. A negative experience is added to S-CA$_2$. However, the certification R-CA$_1$ → S-CA$_2$ was approved during prior observations, thus no negative experience is added to R-CA$_1$.

(d) The chain R-CA$_2$ → S-CA$_3$ → EE$_4$ is obtained. The certificate of EE$_4$ is rejected. Thus, the certificate R-CA$_2$ → S-CA$_3$ must be checked. Assuming its reconfirmation, a negative experience is added to S-CA$_3$, while a positive experience is added to R-CA$_2$.

(e) The chain R-CA$_1$ → S-CA$_2$ → S-CA$_3$ → EE$_5$ is obtained. The certificate of EE$_5$ is accepted. A positive experience is added to S-CA$_2$ and S-CA$_3$. R-CA$_1$ → S-CA$_2$ was evaluated during prior observations, no new experience is added.

Fixing the Key Legitimacy

Different from the issuer trust, which might change over time, key legitimacy theoretically is constant once it is approved. From that point on, the issuer trust in superordinate CAs is of no further relevance. To consider this fact in the trust validation, key legitimacy is set $o_{kl} = (1, 1, 1)$ as soon as enough evidence for the key legitimacy of a trust assessment is available. The question is, when enough evidence is available. One approach is to set the key legitimacy based on the number of positive experiences, or on the number of certificates contained in the trust assessment. To determine the best approach is due to future work.

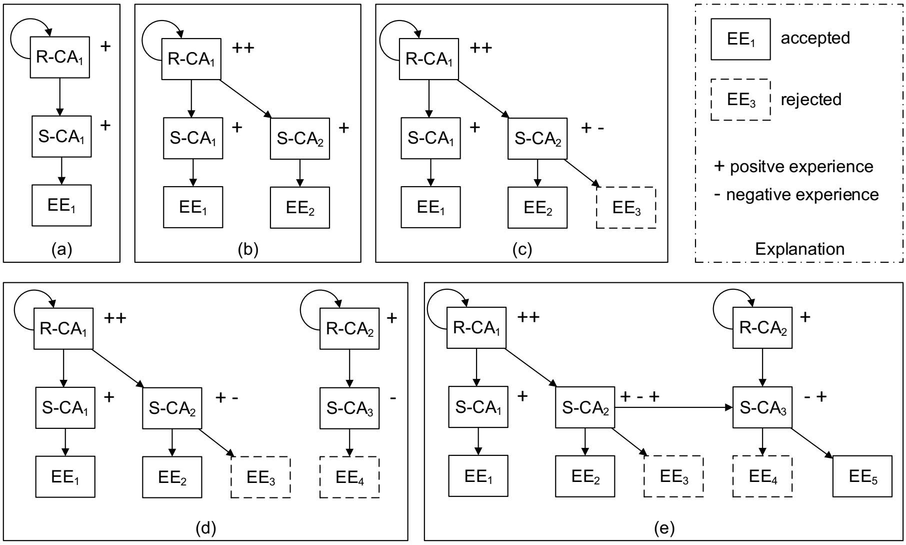

Fig. 1. Evolution of the trust view

Cleaning the Trust View

To prevent a continuous growth of the trust view and to allow the adaptation to current requirements (e.g., changing browsing behavior), a removal mechanism is integrated. A trust assessment *TA* is removed from the local trust view after a fixed time period has been passed since *TA* was last used within trust validation. The length of this time period can be implemented as a system parameter, e.g., one year. The determination of the optimal parameter setting is due to future work.

5 Evaluation

To evaluate our concept, we first summarize the attacker model: First, we assume the user system not to be compromised in the sense, that an attacker cannot manipulate the trust view. Attacking user systems is out of scope of this paper. Second, we assume, that CAs themselves are in general not malicious on purpose. However, a CA's key can be used by an attacker to issue malicious certificates by compromising the CA's key, compelling the CA or by a CA failure, i.e. when the CA issues a certificate to an entity without properly checking the entity's identity. We further assume that once the malicious certificate is detected, countermeasures are taken like revocation and blacklisting of the certificate.

Attacking a CA does not end in itself but aims at attacking secure connections between users and web servers. In general the attacker either aims at a specific user group, i.e. he tries to eavesdrop, monitor or manipulate the communication of a specific group of people or the attacker aims at a specific service, i.e. he tries to attack the communication with a specific web server.

5.1 PKI Attacks

There are two cases, when a user obtains a certificate. Either, trust validation succeeds. In this case, the user solely relies on the CA system. Yet it implies, that the user has seen the certificate before, or the user has sufficiently many good experiences with the involved CAs. The requirements thereby increase with the security criticality of the application. Or, trust validation fails, in this case the validation services are queried to reconfirm the certificate. Given the certificate is trustworthy, the first case will occur most of the times (users in general stick to a limited set of CAs [1] as they repeatedly access the same services, and mostly web servers stick to the same CA when renewing their certificates). Thus, it is acceptable to have costly reconfirmation procedures as e.g. querying several independent notaries, and we may assume, that the validation services do not jointly provide false reconfirmations. This implies, that the attacker must compromise a CA which has a high reputation in order to be successful. However, this leads to several disadvantages for the attacker:

If he wants to attack a specific user group he must compromise a CA, with a high trust level in each of the user's trust views, otherwise the attack will be detected as the certificate is checked with the validation services by those users where trust validation fails. As the trust views are user-specific and not publicly visible, it is hard to identify such a CA trusted by all the users and it is questionable if the attacker can even manage to compromise that CA. Thus, the group of users where an attacker would be successful is reduced to a smaller group, which is unknown to the attacker and furthermore, the time span where a successful attack is possible is reduced due to the increased probability of being detected. The disadvantages for the attacker thereby grow with the number of the applications he tries to attack and their respective security levels. An attacker might also try to attack several CAs, but besides the increased difficulty to attack more than one CA at a time, he is not able to distinguish which user is to be attacked with which malicious certificate.

If the attacker aims at a specific service, he has the same problem of identifying a CA which is sufficiently trusted by all attacked users in order not to trigger the validation services and thus risking a fast detection. Besides that, users that used the service before have pinned the web server's certificate, which further increases the probability of a detection. The CA with which the attacker certainly has the highest success probability, is the CA that issued the certificates for the web server in the past. Again, it is in question, if the attacker can manage to compromise a specific CA.

In summary, this shows, that by the use of trust views an attacker can hardly employ accidental CA failures. The possible damage is reduced due to the limitation of the number of attackable users, while the attacker has a limited and unspecific choice of CAs. Furthermore, the damage a possible CA compromise may cause, highly depends on the CAs visibility in the certification business, which is a much more natural setting than each existing CA being equally critical. Furthermore, it becomes easily detectable which CAs need especially strong protection.

5.2 Attacks on Computational Trust

The trust views of the users govern when validation services are to be queried. Thus, attackers can try to attack the computational trust model in order to improve their success probability when employing a malicious certificate. The aim in this case is to disturb the correct functioning of the trust based decision processes. We discuss the standard attacks on computational trust, and how they apply to our system.

Whitewashing: This means, an entity (in our case a CA) re-appears in a system under a new identity to get rid of negative reputation. In our scenario this implies a CA that issued many incorrect certificates in the past. However, for a CA to re-appear in the system under a new identity and with a new key, there are significant hurdles. The CA's key needs to be certified by another CA which is already part of the Web PKI, or the CA needs to be incorporated into trusted root stores in order to pass path validation. Thus, whitewashing is prevented by standard mechanisms like audits or the required approval by a non-malicious CA, which a malicious CA is not likely to pass without essential changes in its processes and structure. This on the other hand can then justify such a whitewashing of the trust values. After re-appearing, before being trusted by users, the CA needs to demonstrate trustworthiness by correctly issuing certificates, which also requires to be chosen as a CA by web page operators.

Sybil attacks: A sybil attack means, that an attacker of a reputation system forges or controls other entities to produce many good ratings for a certain entity. Yet, such an attack has only limited relevance to our system as there are no unauthenticated entities that provide recommendations. An adversary could mount a sybil-like attack by attacking the underlying validation services in order to maliciously reconfirm certificates by a certain malicious CA and thus falsely improve the reputation of that CA. Yet, in this case, the attacker can directly exploit the malicious reconfirmation and therewith annul the trust evaluation. This shows the importance of the security of the underlying validation services. However, a successful attack requires both, the compromise of a CA and of the several validation services.

Exploiting slow trust adaptation: Such an attack means, that the good reputation of a CA can be exploited by an attacker, as it takes some time before the CA's good reputation is adapted when a sudden incident changes the CA's trustworthiness. Our system does not prevent such attacks. Thus, a CA that is used by many web pages and that built up a good reputation is a major goal for attackers. However, the possible loss of the good reputation and their public visibility provides strong incentives to such heavyweight CAs to put strong protection mechanisms in place.

In summary, for an attacker to benefit from attacks on the trust model, long term planning is required. Furthermore, additional mechanisms must engage which the attacker cannot influence or control, as for example web page operators must actually employ a CA in order that the CA becomes visible and trusted within the users' trust views.

5.3 Limitations

While trust views can significantly lower the risk of relying on a malicious CA, local experiences are no guarantee for correctness. Trust views do not protect CAs from being compromised. If a CA, for which many positive experiences were collected, suddenly fails, the relying entity may still falsely rely on a malicious certificate issued by such a CA. Yet, a CA compromise only threatens those entities, that trusted in the CA before the compromise, which limits the benefit for attackers. On the other hand it is also possible, that a connection is falsely evaluated not to be trustworthy, which relies in the nature of basing decisions on incomplete information.

Furthermore, trust in the key legitimacy of a service provider's key is different from the trustworthiness of the service provider itself (which, for example, comprises the quality of the web page and its contents provided by the service provider). The latter is not addressed by the trust view concept and requires additional mechanisms like the Web of Trust[1] or commercial web page ratings[2]. However, such mechanisms require authentication, which is achieved via authentic public keys.

6 Conclusion and Future Work

We have presented the concept of trust views. The trust view maintains a minimal set of trusted CAs and furthermore assigns different ratings to each CA, such that trust decisions can be made depending on the context. Thus, the risk of relying on a malicious certificate can be governed by the assignment of adequate security levels to the applications. This enables more restrictive trust decisions for critical applications like e-banking, where security is more important and less restrictive rules for less security critical applications. These rules can be adapted to the relying entity's risk profile.

Due to the user-specific reduction of the total number of trusted CAs, the possible attack surface for CA attacks that threaten the respective entity is reduced. Compromises and misbehavior of CAs that are not included in the trust view have no effect as such certificates always require additional reconfirmation and are untrusted when in doubt. CAs that have often been observed—and most probably will also often be observed in the future—achieve higher issuer trusts than CAs that are barely observed. This provides a trade-off between trusting in (a limited set of) CAs and the costly reconfirmation of certificates. Additionally, privacy problems are mitigated as validation services are only queried in rare cases and not for every connection establishment, which allows user profiling in the long run. Furthermore, the load for validation services such as notary servers is reduced. The application of certificate pinning, i.e. storing trusted certificates, further mitigates the threat of relying on malicious certificates for previously

[1] `https://www.mywot.com/`

[2] e.g., Norton SafeWeb `https://safeweb.norton.com/` or McAfee SiteAdvisor `http://www.siteadvisor.com/`

accessed services and prevents repeated reconfirmations. On the other hand, trust validation allows the automated exchange of stored certificates. Considering the user involvement, which is known to be problematic, the trust view concept also allows a fine grained steering: the user is only involved when the local knowledge and the data of the validation services is not sufficient for a decision. Thus, the total number of warnings is reduced and mainly limited to highly security sensitive services, which addresses the problem of warning fatigue.

Currently, we are working on the implementation of the presented concept. Challenges follow from the configuration adjustment to adapt an individual trust view to the needs of it's owner and to balance the system parameters. Thereby, the reduction of false decisions plays an important role. Furthermore, we will realize possible extensions. The model allows to combine local information with expert recommendations based on different indicators of trustworthiness or recommendations from other users. Challenges are the prevention of false or malicious recommendations and the authentication of the recommenders.

Besides that, a trust view allows to detect anomalies, like the unanticipated exchange of a locally stored certificate. We aim at making this local knowledge applicable for compromise detection.

References

1. Braun, J., Rynkowski, G.: The potential of individualized trusted root stores: Minimizing the attack surface in the light of ca failures. Cryptology ePrint Archive, Report 2013/275 (2013), http://eprint.iacr.org/
2. Carnegie Mellon University. Perspectives Project, http://perspectives-project.org/ (visited July 2012)
3. Chadwick, D.W., Basden, A.: Evaluating trust in a public key certification authority. Computers & Security 20(7), 592–611 (2001)
4. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (2008)
5. Eckersley, P., Burns, J.: The (Decentralized) SSL Observatory. Invited talk at 20th USENIX Security Symposium (August 2011)
6. The EFF SSL Observatory, https://www.eff.org/observatory
7. Ellison, C., Schneier, B.: Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. Computer Security Journal 16(1), 1–7 (2000)
8. Evans, C., Palmer, C., Sleevi, R.: Public Key Pinning Extension for HTTP. Internet-Draft (2013)
9. Gutmann, P.: Pki: it's not dead, just resting. Computer 35(8), 41–49 (2002)
10. Gutmann, P.: Engineering Security (2013), Book draft available online at http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf
11. h online. Flame – oversights and expertise made for Windows Update worst case scenario, http://h-online.com/-1614234 (visited July 2012)
12. h online. Fake Google certificate is the result of a hack, http://h-online.com/-1333728 (visited November 2011)
13. Herley, C.: So long, and no thanks for the externalities: the rational rejection of security advice by users. In: Proceedings of the 2009 Workshop on New Security Paradigms Workshop, NSPW 2009, pp. 133–144. ACM, New York (2009)

14. Holz, R., Braun, L., Kammenhuber, N., Carle, G.: The ssl landscape: a thorough analysis of the x.509 pki using active and passive measurements. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 427–444. ACM, New York (2011)
15. Huang, J., Nicol, D.: A calculus of trust and its application to pki and identity management. In: IDTrust 2009, pp. 23–37. ACM, New York (2009)
16. ICSI. The ICSI Certificate Notary (2013), `http://notary.icsi.berkeley.edu/`
17. Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9, 279–311 (2001)
18. Jøsang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Electronic Commerce Conference (2002)
19. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43, 618–644 (2007)
20. Jøsang, A.: An algebra for assessing trust in certification chains. In: Proceedings of the Network and Distributed Systems Security Symposium (NDSS 1999). The Internet Society (1999)
21. Marlinspike, M.: Convergence, `http://convergence.io/` (visited July 2012)
22. Maurer, M.-E., Luca, A.D., Kempe, S.: Using data type based security alert dialogs to raise online security awareness. In: SOUPS, p. 2 (2011)
23. Maurer, U.M.: Modelling a Public-Key Infrastructure. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 325–350. Springer, Heidelberg (1996)
24. PSYC. Certificate Patrol, `http://patrol.psyced.org/`
25. Ries, S.: Extending bayesian trust models regarding context-dependence and user friendly representation. In: Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 1294–1301. ACM, New York (2009)
26. Ries, S., Habib, S.M., Mühlhäuser, M., Varadharajan, V.: Certainlogic: A logic for modeling trust and uncertainty (short paper). In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 254–261. Springer, Heidelberg (2011)
27. Ruohomaa, S., Kutvonen, L., Koutrouli, E.: Reputation management survey. In: Seventh International Conference on Availability, Reliability and Security (ARES 2007), pp. 103–111 (2007)
28. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying wolf: An empirical study of ssl warning effectiveness (2009), `http://static.usenix.org/event/sec09/tech/full_papers/sunshine.pdf`
29. Wazan, A.S., Laborde, R., Barrère, F., Benzekri, A.: A formal model of trust for calculating the quality of x.509 certificate. Security and Communication Networks 4(6), 651–665 (2011)
30. Wazan, A.S., Laborde, R., Barrère, F., Benzekri, A.: The x.509 trust model needs a technical and legal expert. In: ICC, pp. 6895–6900 (2012)
31. Weaver, G.A., Rea, S., Smith, S.W.: A computational framework for certificate policy operations. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 17–33. Springer, Heidelberg (2010)
32. Zhang, Y., Hong, J.I., Cranor, L.F.: Cantina: a content-based approach to detecting phishing web sites. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 639–648. ACM, New York (2007)
33. Zimmermann, P.R.: The official PGP user's guide. MIT Press, Cambridge (1995)

A User-Centric Digital Signature Scheme

Felipe Carlos Werlang[1], Ricardo Felipe Custódio[1], and Martín A.G. Vigil[2]

[1] Laboratório de Segurança em Computação,
Universidade Federal de Santa Catarina,
Caixa Postal 476, 88040900 Florianópolis, Brazil
{felipewer,custodio}@inf.ufsc.br

[2] Technische Universität Darmstadt,
Hochschulstr. 10, 64289 Darmstadt, Germany
vigil@cdc.informatik.tu-darmstadt.de

Abstract. We observe that current mainstream digital signature schemes are complex and inconvenient for end users. We group the main problems related to these schemes and propose a new approach, centered on the needs of the end user. The new model is a redesign of the overall signature process, discarding certificates and the X509 PKI in favor of simple structures and natural trust relationships modeled on conventional handwritten signatures.

Keywords: Digital signature, user-centric, usability, notary, notarization, long-term maintenance, trust.

1 Introduction

Digital signatures are an essential technology in enabling the shift from paper to paperless environments. The use of electronic documents brings substantial gains in terms of efficiency and savings. This has led several countries to create specific legislation regarding the legal validity of digital signatures and electronic documents. Examples include the United States [1,2], the European Union [3] and Brazil [4]. These legislations have favored digital signatures schemes reliant on Public Key Infrastructures (PKIs), especially the X509 [5] PKI model.

Unfortunately, PKIs have well known deficiencies. Ellison and Schneier warn in [6] about the main risks of PKI from a security perspective. Lopez et al. describe in [7] the main technical, economical, legal and social reasons that led to failures in the employment of PKI in different areas. Current real-world business demands have to be adapted and constrained to work within the PKI model [8]. Digital signature schemes are no exception.

In our country there are two large hierarchical PKIs. One is an educational PKI [9] managed by universities and research institutions and used as a test bed for technology development. The other one is the national PKI [10], run by the government. It was established in an effort to promote the nationwide use of digital signatures. Therefore, it's main purpose has been supplying the digital certificates needed for signing electronic documents. However, a decade later,

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 152–169, 2014.

digital signatures are mostly in use within governmental institutions and private companies. Their use for personal purposes remains negligible.

One important factor for the limited adoption of digital signatures is the acquisition cost of certificates and cryptographic devices. In contrast to paper documents and handwritten signatures, the use of digital signatures is only cost-effective in environments with high signature volumes. However, other aspects like the challenges for long-term signature preservation and the overall complexity of the signature process play an even bigger role.

From our experience, we have concluded that, currently, the apparatus necessary to create a digital signature is far too complex and inconvenient for the end user. Moreover, the model in place does not adequately reflect the natural interactions and trust relationships between users. We provide more details to corroborate this claim in Section 2.

Therefore, we propose a different approach to digital signatures, one that is centered on the user's necessities. Objectively, a user shall be able to create a digital signature easily and quickly, without the requirement of any prior registration process. Additionally, it shall be possible for a user to assure the trustworthiness of a signature later, with the help of a trusted third party of his choosing. Our approach also improves the long-term signature maintenance and provides a natural trust model based on the relationships established between entities.

The approach is inspired by the handwritten signature model used in the countries that practice the *Civil Law Tradition* [11]. Nevertheless, the main concept, i.e., applying a personal mark to a document and having a third party or government official authenticate the mark when needed, can also be observed in other cultures. The Japanese culture, where seals (*Hanko* or *Inkan*) replace handwritten signatures, is such an example.

This paper is structured as follows. In Section 2 we analyze the main problems concerning the use of digital signatures in conjunction with X509 PKIs. Next, in Section 3 we present the related work. Then, in Section 4 we present our user-centric digital signature scheme. In Section 5 we analyze how our proposal addresses the problems described in Section 2 and how it compares to related work. Finally, in Section 6 we draw our conclusions and discuss possibilities for future work.

2 Problems

To create a *hand-written signature*, one simply draws his personal graphic mark on the paper document. Then, depending on the purpose of that document or on the trust relationship between the signatory and the receiver of the document, a third party authentication may be required. In other words, if the receiver does not trust the signatory, he may require a trusted third party attestation to the link between the graphic mark and the identity of the signatory. The attestation is known as *notarization*, since it is normally performed by a notary.

A digital signature also identifies the signatory of a document. But this identity is only reliable if the validity of the signatory's public key can be verified. This is achieved with the use of digital certificates, which are issued by trusted parties called Certification Authorities (CAs). A CA authenticates public keys and key holders, and signs certificates asserting that a public key indeed belongs to who claims to have the corresponding private key. A CA signs certificates for either end users or other CAs. The infrastructure to support the use of certificates is called Public Key Infrastructure (PKI) [12].

In contrast to handwritten signatures, the authenticity provided by a digital signature is not everlasting. This is because the trustworthiness of the corresponding public key and the security of cryptographic algorithms are not everlasting. A public key is trustworthy as long as the corresponding certificate is valid. The validity of a certificate ends when: (i) the validity period defined by the issuing CA ends or (ii) the issuing CA revokes the certificate. The cryptographic algorithms necessary to create digital signatures become insecure as computer power and cryptanalytic techniques evolve.

The extension of authenticity beyond the lifetimes of certificates and cryptographic algorithms requires the use of timestamps. Since timestamps are signed objects, their authenticity fades out too. Additionally, the authorities issuing the timestamps also have to be trusted. In the end, everlasting authenticity requires an endless use of timestamps, an ever-increasing number of trusted entities and an ever-growing accumulation of *cryptographic evidence* (certificates, revocation statuses and timestamps).

Furthermore, while the prerequisites for a conventional signature do not exceed a regular pen and a piece of paper, a digital signature requires the possession of a digital certificate. To obtain a digital certificate one normally has to follow a registration process at a Registration Authority (RA), which then contacts a CA to issue the certificate. Depending on the service provider, this process can take days. In addition, the concept of the certificate could be compared to a driver's license. One gives the permission to drive and the other to sign electronic documents, except for the fact that you should not really need a license to sign something, nor should you need to buy a new one after a couple of years.

Then there is the problem of key storage. The certificate is tied to a specific private key that has to be stored in some type of cryptographic device or as an encrypted file. Therefore, if it is in a device, you have to carry that specific device with you all the time, or foresee the moment in which you will have to sign something. If it is stored in a computer, you will only be able to generate signatures from that computer.

Finally, there is the discrepancy in the notion of trust in the real world and the one imposed by certificates. Normal trust relationships are bilateral. They are established over time, in a slow fashion, and are based on experiences. Nevertheless, this relationship can be rapidly destroyed. On the other hand, certificates require unilateral trust in a third party (i.e. the CA) and the trust in a given certificate is imposed, without the existence of prior experiences [7]. Additionally the trust relationship with the third party cannot be terminated.

Given this prospect, we enumerate four main problems:

Problem 1. The older the document signature, the higher the storage and processing overhead. The reason is that: (i) the accumulation of cryptographic evidence requires an ever-increasing storage space, and (ii) the validation of the document signature requires, in turn, the validation of the signature of each accumulated cryptographic evidence.

Problem 2. Trusting in former trusted parties is an issue. A party that was trusted in the past may disappear, leaving no data for future signature verifiers to assess the party's trustworthiness. It may also be the case that the party does not fulfill the necessary requirements in the future [13].

Problem 3. The complexity and design characteristics of the current digital signature scheme make the signature generation inconvenient for end users. Moreover, both signatory and receiver need to be aware of the factors that affect the authenticity of a signature in the long-term. The absence of such awareness can result in losses for one or both parties in the event of a signature invalidation.

Problem 4. The notion of trust provided by certificates is not the same as the notion of trust in the real world which is based on relationships and experiences made over time [7].

3 Related Work

Improving the techniques used for maintaining the authenticity of document's signatures in the long-term has been the focus of many proposals found in the literature. Many of those rely on a trusted third-party to certify a particular aspects of a signature, thus being characterized as forms of notarization. The *Cumulative Timestamps* technique, proposed in [14] and [15], is such a case. It has become a prominent technique for extending the lifespan of a digital signature. Although ultimately impractical, as shown in Section 2, it is currently the strategy recommended in digital signature standards like CAdES [16] and XAdES [17].

Cumulative Notarizations [13] is a similar proposal. In this approach a notary certifies the authenticity of a signature or the authenticity of previous notarizations once they near the end of their validity period. Problem 1 is addressed because only the cryptographic evidence related to the last notarization is kept. The proposal also addresses Problem 2 because trust is shifted from one notary to another notary at each new notarization. Thereby former trusted parties become irrelevant.

In contrast, in the *Optimized Certificate* [18] proposal, only cryptographic evidence is certified by the trusted third-party. The idea here is that the cryptographic evidence is sent to, and verified by, a special purpose certification authority. Next, the authority issues an *optimized certificate* which replaces all cryptographic evidence. An optimized certificate is a smaller digital certificate containing the same identity data as the original signatory certificate, but its validity is restricted to the moment of emission. This certificate also serves as a timestamp. Optimized certificates can also be renewed once the cryptographic algorithms used in their emission are close to becoming insecure. The proposal addresses both Problems 1 and 2. Nevertheless, it is impractical since certificate substitutions are prevented in current advanced electronic signature formats.

Finally, Vigil et al. propose in [19] the use of Notarial Authorities (NAs) as the central part of a new type of PKI, designed for documents signatures. It is called Notary Based PKI (NBPKI). In the NBPKI, the trusted parties assert if a document's signatory certificate is valid to verify a particular document signature at a certain date and time. Document signatories and NAs generate their key pairs and sign their own X.509 certificates. Signatories register their certificates at Registration Authorities (RA). A document's signatory certificate is only trustworthy if notarized. A verifier submits the document signature and the signatory's certificate to an NA. The NA checks the validity status of the certificate with one or more RAs. If the certificate is currently valid, the NA returns a signed assertion of the validity of the certificate and the existence of the document signature at the current date and time. The verifier has only to trust in the issuing NA's public key. The maintenance of document signature consists of replacing the current assertion by a new assertion. The NBPKI properly addresses Problems 1 and 2. Problem 3 is partially addressed.

4 User-Centric Digital Signatures

A user-centric digital signature scheme means that the signature scheme is designed to best suit the needs of the end user. It is an attempt at a redesign, from the ground up, of the overall digital signature process. The scheme is inspired by the conventional handwritten signature model practiced in most of Europe and Latin America. Nonetheless, other cultures can relate to it as well. Therefore, the entities involved, as well as the interactions between them, are intended to be familiar to most users.

The goal of the proposal is in empowering the end users, namely the Signatory and the Receiver (hereinafter called Verifier) of a given signed document. This means that the users are the ones who choose the requirements of a specific signature, based on the relevance of the signed document. Likewise, they choose who to trust and for which purposes.

In the most basic form, the signature process consists of the signatory signing a document with his private key and passing the document along to the verifier. At this point, the verifier has no guarantee on the signature's authenticity. Therefore, he resorts to a third party (i.e., a Notary) of his trust to authenticate the signature. In practice, either the signatory or the verifier can obtain the signature authentication (hereinafter called notarization) from the selected notary. But it is in the verifier's right to choose which notary shall be used. The signatory also has to register his attributes and his public key at that notary's establishment if he hasn't previously.

The notarization consists of the notary's signature upon the original signature and the current date and time. In the case of *Full Notarizations* (cf. Definition 2), the signed document is also encompassed. Thus, a notarization serves as both an assertion of authenticity and a timestamp. As a result, if compared to the X509 PKI, a Notary combines characteristics of both Registration, Certification and Time Stamp Authorities (RA, CA and TSA respectively).

The remainder of this section starts with the description of the entities involved (Section 4.1) and the assumptions that ground the proposal (Section 4.2). Next, we present our definitions (Section 4.3) and procedures (Section 4.4).

4.1 Entities

The proposal is comprised of four entities: end users, namely the Signatory and the Verifier; the Notary and the Scheme Operator.

Signatory: The Signatory is a user that creates a signature for a given document. He is responsible for generating his own key pair and for registering the public key at a notary establishment.

Notary: The Notary is a trusted party. He is responsible for user enrollment, i.e., registering users' attributes and public keys, and for issuing notarizations. Each notary has a physical office for enrollment and an online autonomous notarization service. Notaries are listed as Trusted Service Providers (TSPs) in the Trusted-service Status List (TSL) [20].

Scheme Operator: The Scheme Operator is the body responsible for the management and publication of the TSL. This task may be embodied, for example, by a government agency or a specific department within an organization. In order to be listed in the TSL, TSPs will have to be assessed. The criteria and methods used in this evaluation may vary depending on legislation and organizational policies.

Verifier: The Verifier is the end user interested in assessing the *authenticity* (the origin can be identified), the *integrity* (the content has not been altered) and *chronological proof of existence*[1] of a given signed document.

4.2 Assumptions

The proposed scheme is grounded on the following assumptions:

Assumption 1. The Notary is a trusted entity within the socio-economic system, either on a national or organizational level. He must abide by laws and regulations concerning a regular notary service [13].

[1] Also known as *proof of existence*, this property provides the date and time at which an object existed[21]

Assumption 2. Notary establishments are well distributed geographically. End Users should be able to perform enrollment or key pair substitution procedures within the same city or organizational unit.

Assumption 3. The notary establishment has a *Secure Storage* to store End Users' data, signed documents and notarizations. This storage must offer integrity, authenticity and secrecy guarantees upon stored content. It should also be properly equipped with backup contingencies. Additionally, its security must be independent of the security of the Notary's key pair.

4.3 Definitions

Definition 1. *A Signature is a tuple $S = (h, A, pk, \sigma)$, where h is the hash of the signed document, A is a set of the signatory's attributes where $A \neq \emptyset$, pk is the signatory's public key, σ is the signatory's signature on the concatenation $h||A||pk$, using his private key sk.*

Definition 2. *A Full Notarization is a tuple $FN = (id, v, y, f, t, nk, \delta)$, where id is a unique identifier for FN, $v > 0$ is the current version of the notarization, y is the hash of the concatenation $D||S$, where D is the signed document and S is the signature (cf. Definition 1), f is the time of the first notarization, i.e., $v = 1$, t is the time when δ was created, nk is a unique identifier for the Notary's public key, δ is the Notary's signature on the concatenation $id||v||y||f||t||nk$. For $v = 1$, $f = t$.*

Definition 3. *A Partial Notarization is a tuple $PN = (id, v, y, W, f, t, nk, \delta)$, where id is a unique identifier for PN, $v > 0$ is the current version of the notarization, y is the hash of the concatenation $h||S$, where h is the hash of the signed document and S is the signature (cf. Definition 1), W is a set of instances of h using different hash algorithms where $|W| = v$, f is the time of the first notarization, i.e., $v = 1$, t is the time when δ was created, nk is a unique identifier for the Notary's public key, δ is the Notary's signature on the concatenation $id||v||y||W||f||t||nk$. For $v = 1$, $f = t$.*

4.4 Procedures

Notary Management: Notary management refers to the tasks performed by the Scheme Operator and it is basically comprised of notary registration and service status changes.

A Notary is responsible for generating his own key pair. Even so, key size and employed algorithms must comply with the Scheme Operator's requirements.

The Scheme Operator registers a Notary by listing him as a Trusted Service Provider (TSP) in the Trusted-service Status List (TSL). He also includes the Notary's public key as the *service digital identity* and sets the service status as "in accordance". The Scheme Operator may later change the service status if needed. Examples of such situations include a temporary suspension or a service revocation. In the latter case, the Notary has to generate a new key pair and provide the new public key to be included in the TSL.

A Notary's key pair has no fixed validity period. Nonetheless, the Scheme Operator may arbitrarily revoke his *accordance* status at any moment. This generally occurs if the Notary's private key is compromised or if the cryptographic algorithms used in key pair generation become insecure. There is also the possibility of the Scheme Operator enforcing a periodical key pair renewal policy.

In the event that a Notary ceases his operations, the service status is changed to "revoked" or "not renewed". In this case, all the stored data regarding the notarizations he performed must be absorbed by another Notary.

Notary assessment, as well as TSL implementation and management are subject to local legislation and/or organizational policies. Nonetheless, existing norms [20] should be respected.

User Management: The user management is performed by the Notary and is comprised of user enrollment and registration updates.

The Notary starts the enrollment process by verifying the documentation presented by the user. He then collects the user's public key and attributes. Attributes may range from personal information like name, date of birth and driver's license number to professional information like roles within an organization and so on. They may contain any other information that the user judges useful, as long as the proper documentation to back it up is provided. The Notary also requires from the user a proof of possession of the private key.

A user registration is updated if there are changes in the attributes or if the key pair needs to be substituted. A key pair substitution occurs if the user's private key is compromised or if the cryptographic algorithms used in its generation are no longer secure. In case of key compromise, the Notary suspends new notarizations for that key as soon as notified by the user. In case of aging algorithms, new notarizations are suspended for all keys generated with those algorithms. Notarizations are resumed with the registration of a substitute key pair. The Notary may also require key pair substitutions periodically or based on the amount of times a key has been used.

The user is allowed to keep multiple keys registered at the same time. Additionally, he can define restrictions on the number of times a specific key can be used.

Signature: The signature process input is composed of the signatory's private key and the result of the message digest calculation process. This message digest is calculated on the concatenation of the following elements: the content message digest, h, a set containing the signatory's attributes, A, and the signatory's public key, pk. A may optionally be composed of just a subset of the signatory's attributes based on their relevance to the content being signed. The resulting signature value, σ, and the signed elements together compose the signature container S:

$$S = (h, A, pk, \sigma)$$

The signatory may use a key pair previously generated or generate a new one at signing time. For the signature to be notarized, the related public key must first be registered at the chosen notary establishment. Once registered, thi key may be used multiple times, i.e., future notarizations don't require new registrations.

Notarization: The notarization is basically a third-party signature that encompasses the signed document, the signature and the current date and time. It serves as both an assertion of validity and a timestamp. As such, it endorses the identity and/or the attributes claimed by the signatory and attests to the existence of the signature prior to a determined point in time.

A Notarization can be either a Full Notarization or a Partial Notarization. The full notarization follows the protocol below:

$$\begin{aligned} \mathcal{U} &\longrightarrow \mathcal{N} : (D, S) \\ \mathcal{N} &\longrightarrow \mathcal{U} : \underbrace{(id, v, y, f, t, nk, \delta)}_{FN} \end{aligned}$$

A user requests a notarization for a signature sending both the signature, S, and the signed document, D, to the Notary. The Notary first identifies the signatory based on the public key present in the signature. If this key belongs to an enrolled user, he checks the claimed attributes and verifies the signature using the user's public key. If the verification is successful a notarization is generated.

The first step to generate the notarization container is the calculation of the content message digest, y, composed of the signed document and the signature. Next, the Notary generates a unique identifier, *id*. Then, he adds the notarization version, v, which in this case must be 1, and the current time, t. Since this is the first notarization on this signature, i.e., $v = 1$, f receives the same value as t. Additionally, the Notary's public key identifier, nk, is included in the container. At last, a signature encompassing all previous elements is generated with the Notary's private key. The signature value δ is also included in the notarization container. Before returning the notarization to the user, the Notary saves it along with the signed document and the signature in his storage.

A Partial Notarization, in turn, is a notarization where the Notary does not have access to the content of the signed document. It follows the protocol below:

$$\begin{aligned} \mathcal{U} &\longrightarrow \mathcal{N} : (h, S) \\ \mathcal{N} &\longrightarrow \mathcal{U} : \underbrace{(id, v, y, W, f, t, nk, \delta)}_{PN} \end{aligned}$$

When requesting a partial notarization, the user sends the document's message digest, h, along with the signature, S, to the Notary. The Notary proceeds basically the same way as in a Full Notarization apart from the following exceptions:

- He does not verify the mathematical correctness of the signature. This would require possession of the signed document;
- He uses h instead of the signed document in the content message digest, y, calculation;
- He includes the set with the signed document's message digests, W, in the notarization container. At this point W is composed of only one element, h;
- Before returning the notarization to the user, he saves only the signature and the notarization in his storage.

Partial Notarizations are better suited for situations where secrecy is required, or for documents with temporary relevance. Maintaining the the authenticity of a partial notarization in the long-term requires the user to be aware of the security status of the cryptographic algorithms involved. This is because the user is supposed to renew the notarization before any of the algorithms becomes insecure.

It should be noted that either one of the users, signatory or verifier, can request a notarization for a signature. The Notarization renewal procedure is described later in this Section.

Signature Validation: To be considered trustworthy, a signature must be mathematically and semantically correct. The mathematical correctness is guaranteed if the signature is valid under the signatory's public key. The semantic correctness is ensured if the binding between the signatory's public key and his attributes was valid when the signature was created. This binding is attested by the notarization, therefore the latter must also be mathematically and semantically correct. Mathematical correctness is guaranteed if the notarization is valid under the Notary's public key. Semantic correctness is ensured if the Notary's service was *in accordance* when the notarization was created and continues to be so at the time the end user's signature is verified.

Given a signature S and a full notarization FN (Definitions 1, 2), the signature validation process is composed of the following steps:

1. *Notary validation*: The verifier uses the public key identifier, nk, from the notarization to check if the correspondent key identifies a Trusted Service Provider in the TSL and if its current service status is "in accordance".
   1.1 *(OPTIONAL) Personal trusted list check*: The verifier uses the public key identifier, nk, to check if the Notary is present on his personal trusted parties list.
2. *Notarization validation:* The verifier verifies the mathematical correctness of the signature value, δ, using the signed document, D, the signature, S, and the signed elements in FN along with the Notary's public key extracted from the TSL as input.

If both validations are successful, the signature is considered trustworthy. There is no need to verify the mathematical correctness of signature value, σ, in S since this verification was already performed by the Notary during the notarization process.

When verifying a signature S with a partial notarization PN (Definitions 1, 3), the signature validation process is comprised of steps 1 and 2 as before and these additional steps:

3. *Hash collision detection:* The verifier calculates digests of the signed document and compares them to the digests in field W. This prevents collision attacks.
4. *User Signature validation:* The verifier verifies the mathematical correctness of the signature value, σ, using the signed document, D, and the signed elements in S along with the signatory public key, pk, as input.

If all validations are successful the signature is considered trustworthy.

Notarization Renewal: Notarization renewals are necessary when: (i) the Notary's accordance status is revoked or (ii) the used cryptographic algorithms are no longer secure or are about to become insecure. In other words, semantic correctness can no longer be asserted.

There are slight differences between the renewal processes for Full Notarizations and Partial Notarizations. The full notarization renewal follows the protocol below:

$$\begin{aligned}\mathcal{U} &\longrightarrow \mathcal{N} : id\\ \mathcal{N} &\longrightarrow \mathcal{U} : \underbrace{(id, v, y, f, t, nk, \delta)}_{FN}\end{aligned}$$

A user requests a notarization renewal by sending the notarization identifier, id, to the Notary. The Notary fetches the related notarization along with the signature and the signed document from his storage. He then calculates the content message digest, y, composed of the signed document and the signature. Next, he generates a new notarization container using the same values from the original instance for id and f. In addition, he increments the notarization version, v, and includes the current time, t, and current Notary public key identifier, nk. Finally, the Notary signs all previous elements with his private key and includes the signature value, δ, in the notarization container. Before returning it to the user, he also substitutes the notarization in his storage.

The partial notarization renewal follows the protocol below:

$$\begin{aligned}\mathcal{U} &\longrightarrow \mathcal{N} : (h_n, id)\\ \mathcal{N} &\longrightarrow \mathcal{U} : \underbrace{(id, v, y, W, f, t, nk, \delta)}_{PN}\end{aligned}$$

When requesting a partial notarization renewal the user sends the document's new message digest, h_n, and the notarization identifier, id, to the Notary. The Notary fetches the related notarization and the signature from his storage. He then verifies if there is at least one document message digest in W that is still secure, i.e., the cryptographic algorithm used is still secure. This verification prevents intentional collision for the signed document. If the verification succeeds, the Notary renews the notarization. The rest of the renewal process follows the same steps as for the full notarization, except for two details:

– He uses h_n instead of the signed document in the content message digest, y, calculation;
– He includes h_n in the signed document's message digests set, W;

It is important to note that a Full Notarization can be renewed at any point, even after the cryptographic algorithms become insecure. A Partial Notarization, on the other hand, must be renewed while the cryptographic algorithm used in the message digest calculation is still secure. Additionally, it is the verifier's responsibility to request notarization renewals.

5 Analysis

In Section 3 we established that Problems 1 and 2 were already addressed in the *Cumulative Notarization* [13], *Optimized Certificate* [19] and NBPKI[19] proposals. However, Problem 3 is only partially addressed in the NBPKI and Problem 4 stands untouched. In the remainder of this section we analyze how our proposal addresses all four problems. We also describe the additional advantages of the proposed scheme. Lastly, we present implementation considerations.

Problem 1: The long-term maintenance of notarized signatures is less costly than that of conventional digital signatures. Additionally, the signature validation process is simplified. We demonstrate this with the following example:

Let's consider 3 distinguished scenarios: (a) A basic single level PKI. This PKI has a single CA issuing certificates for the users and a single TSA issuing timestemps; (b) A multi-level PKI composed of a Root CA with one CA and one TSA under it, in the second level; (c) A notary issuing partial notarizations (c.f. Definition 3) for signatures. Additionally, let's consider that in both (a) and (b) the validity period of timestamps is 5 years and, due to algorithm obsolescence, the notarizations in (c) have to be renewed every 15 years.

In (a), the initial cryptographic evidence needed to preserve the validity of the signature in the long-term is composed of: timestamp, signatory certificate, CA certificate, TSA certificate and signatory certificate revocation status. Then, every 5 years a new timestamp and TSA certificate are added to the package. In (b), the initial cryptographic evidence is composed of timestamp, signatory certificate, CA certificate, TSA certificate, Root CA certificate and revocation statuses for signatory and CA certificates. Every 5 years a new timestamp and TSA certificate are added to the package, plus the previous TSA certificate revocation status. In (c) the only cryptographic evidence stored by the user is a single notarization. It is substituted every 15 years.

Figure 1 illustrates the amount of cryptographic evidence accumulated in each scenario over a period of 40 years. It shows a linear increase in the amount of evidence stored in scenarios (a) and (b), while in scenario (c) only one notarization has to be accounted for. As this notarization is renewed, its size increases because the size of cryptographic material (keys and message digests) increases over time. Nonetheless, the storage space needed in (c) is substantially smaller than in scenarios (a) and (b). The comparison in Figure 1 is also applicable for the signature validation process. Since every cryptographic evidence is a signed object, the verifier has to check the mathematical correctness of every object. Therefore the validation process in scenario (c) is substantially simplified.

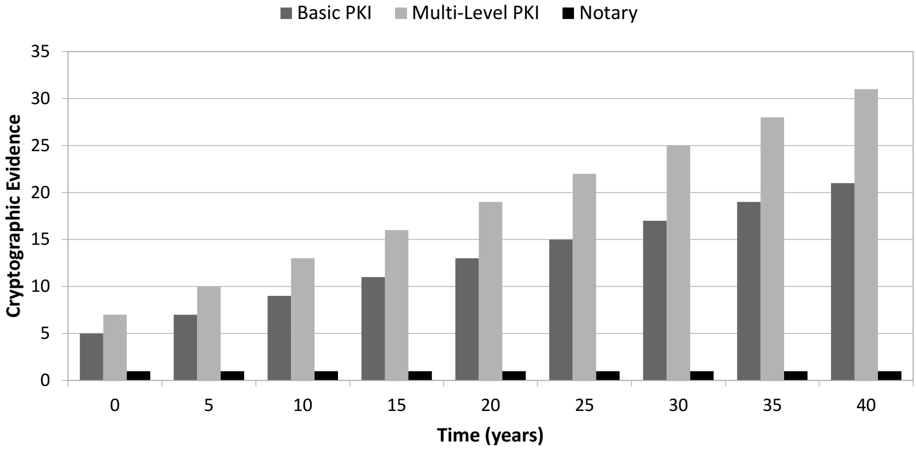

Fig. 1. Cryptographic evidence per signature

For the purpose of simplicity we did not take into account possible storage optimization[2].

Problem 2: The compromise of the notary's private key (which is indicated by the TSL), or the obsolescence of cryptographic algorithms, are reasons for the verifier to renew the notarization. In either case, he only has to trust the notary's key used to sign the last notarization in the signature validation process. Old trusted parties are no longer required in the validation process.

Problem 3: The proposed signature scheme is designed so that a simple or occasional use becomes easy and requires little knowledge. This is illustrated by the example below.

[2] A TSA certificate has a validity period after which a new certificate with a new key pair must be issued. If more than one timestamp is issued by a TSA for the same signature during the lifetime of a certificate, this certificate only needs to be stored once.

Let's take the example of a father and a son. The father gives his son a car, but keeps it registered in his own name. Some time after, the son moves to a new city taking the car along. One day, the son receives an excellent offer for the car, but it is only valid for that day. In this case he needs a proxy document signed by his father in order to transfer the car ownership, but, due to the time constraint, mailing the document is not an option.

Now, assuming the father has no prior experience with digital signatures, let's apply a hypothetical implementation of our proposal: The father goes to a computer and starts by downloading the signer software (or accessing a web-based version). He then selects the proxy document, fills in his name, selects the nearest notary establishment (or the one he normally uses for handwritten signatures) in the list provided by the software and clicks "sign". The software then shows him a code to be written down (or printed) and instructions to go to the notary establishment to complete enrollment. Once at the notary establishment, he presents his personal documents and the code he received to the notary. Back home, the father accesses the signer software, selects the signed document and clicks "notarize". He then only needs to fill in his son's email address and click "send".

In this process, many things happened without the father being aware of them. First, the software has downloaded the latest TSL version. Then, when the father clicked on sign, a keypair for single use was generated, the signature was created, an enrollment request with the public key was sent tho the notary's online service and it, in turn, sent back a proof of possession challenge encrypted with this public key. Next, the signer software decrypted the challenge with the private key and showed the resulting code to the father. When the father clicked "notarize", the signature was sent to the notary service, which automatically verified it, notarized it and sent it back. Finally, the private key used in the process was destroyed, eliminating any need for further key management and revocation.

Since the signature verifier in this example will be the government employee responsible for overseeing vehicle ownership transfers, any notary recognized by the government, i.e., present in the TSL, will do.

The key point in this example is that the user is able to sign a document easily, without having to learn about certificates, new authorities, or even that he has a cryptographic key. In other words, the complexity is hidden within the applications. Arguably, something similar could be achieved for the signature process with PKI, but then the user would have to learn about the inner details later in order to understand why his signature has lost its validity after some time.

In the case of more experienced users, or ones with a regular need for digital signatures, the model enables them to use the same key multiple times, and even register multiple keys for different purposes and with restrictions on the number of times a key is used. Small business are such an example. They may establish who are the notaries in which each of their main clients and suppliers trusts and then register keys at those notary establishments. In small towns where there is only one notary the scenario becomes even simpler. Another possible application is to register a single use backup key for emergencies. For example, a user has a "frequent-use" key registered and the private key is stored in his home computer. Then, while traveling overseas, he has to sign something urgently. In this case he can use the backup key that he carries encrypted with a password in his cellphone. Therefore, only the one notarization will be generated and the private key can be destroyed.

In regards to signature maintenance, the use of the Full Notarization (Definition 2) provides the easiest maintenance. Verifiers don't need to be aware of the security of the algorithms used in the signature process because a full notarization can be renewed at any point. Software performing a signature validation can even request a notarization renewal without prompting the user. Partial Notarizations (Definition 3) have to be renewed before cryptographic algorithms become insecure. Thus, the end user has to be more aware of the underlying complexity of the process than with the full notarization. Nevertheless, this renewal process could be automatically managed by specialized archiving software, as long as it stays online all the time to get updates on the security of the algorithms and to request renewals from the notaries.

Problem 4: Trust in the PKI has a boolean characteristic. Either you trust the Root CA, and by consequence everything that is under it, or you don't trust it. Our proposal, on the other hand, enables different degrees of trust for different entities. Therefore, the user is able to directly relate trust and risk. For example, a construction company may have an internal policy in which it accepts signatures on contracts for small projects notarized by any notary listed on the TSL. But signatures on contracts with its big suppliers have to be notarized by a specific notary with which the company has long lasting business relation. This notary specializes in digital signature notarization, employing, therefore, staff and security technologies equivalent to a large Certification Authority.

This trust model is closer to the real-world trust model, based on existing relationships and experiences. If the proposal is implemented within a closed organization, there already exists a trust relationship between the user and the organization constituted by an employment contract or membership agreement. If it is implemented on a nationwide scale, the user may choose to trust any notary recognized by the government (listed in the TSL), or he may trust only specific notary's with which he has prior experience or which posses an impeccable reputation. And he may also trust one notary in favor of another for distinct purposes depending on the risk of financial losses involved with the document.

In general, the key difference in the proposed approach is that trust is no longer an imposition, but a choice.

5.1 Additional Advantages

Notarized digital signatures offer some additional advantages if compared to traditional signatures relying on X509 PKIs. We describe these advantages below:

Effective Key Revocation: Once the user requests the revocation of his key pair, the Notary immediately stops issuing notarizations for signatures made with that key pair. In other words, no more authentic signatures are generated. Meanwhile, certificate based signatures can still be generated after the certificate is revoked. The revocation effectiveness, in this case, depends on everybody getting the updated certificate revocation status and applying it correctly in the signature validation, which is hard to guarantee.

Dynamically Specifiable Signatory Attributes: The signatory attributes, i.e., A (Definition 1), contained in each signature can be chosen specifically by their relevance to the content and/or goal of signed document. For example, some kinds of documents could require only the signatory name while others would require numbers and addresses or even authorization data like organizational role and membership status.

5.2 Implementation Considerations

Assumptions 2 and 3 bring some cost considerations and implementation challenges that we discuss below.

A notary establishment requires some office space, extra staff in case of high demand and specific equipment, including servers, cryptographic devices, storage solutions and contingency equipment. In countries with the civil law tradition, these establishments are already in place and accordingly staffed. Therefore, only additional equipment and training are required there. In countries where there is no such infrastructure, or if the proposal were to be employed within a closed environment, i.e., a private company, the impact would be bigger. Nevertheless, the establishment of a X509 PKI also requires a substantial investment. And a Registration Authority also has to distribute offices to serve the public. Additionally, the level of security requirements for Certification Authorities is related to their importance, i.e., the security in a CA at the bottom of the hierarchy tends to be milder in contrast to a Root CA. The same applies to notary establishments. The larger the operation coverage and the number or clients or the kinds of business these clients operate in, the higher the demand for security.

In regards to the storage solution, although secure storage may be challenging to implement, as existing notary establishments start to migrate to a paperless environment it is natural to expect that they will have to acquire such capabilities anyway. Other services performed by notaries also require documents to be kept for long periods and in a safe manner. The same applies to courts, governmental institutions and corporations. Additionally, the level of security of the stored data can be a marketing advantage. Vigil et al. survey in [22] the existing approaches for long-term archiving in regards to authenticity, integrity and proof of existence. Approaches for long-term confidentiality are covered in [23].

6 Conclusions

Digital signature schemes reliant on X509 PKIs are complex and inconvenient for end users. Long-term signature maintenance is costly, both in terms of storage and processing power. Furthermore, the trust model required by these schemes does not relate to the way trust relationships are normally established.

We address all these problems by redesigning, from the ground up, the entire digital signature process. This proposal is centered on the end user, making the signatory and the verifier free to define the requirements of a particular signature and to establish their own trust relationships. This comes in contrast to traditional digital signature schemes with a top down approach, where requirements and trust are imposed. In comparison to current digital signatures, our model comes closer to the conventional signature practices culturally established in a large part of the world.

The adoption of this proposal requires the elaboration of a new set of standards, the implementation of new software solutions and a revision of the legal aspects of digital signatures. This requires a great effort, but it is the view of the authors that this will bring better results than continuing to mend a complex infrastructure that was proposed for a different reality than our current one.

Future work includes the implementation of application prototypes for a test environment, followed by the conduction of an efficiency and usability evaluation with real users as a proof of concept.

References

1. United States: Electronic signature in global and national commerce act ("E-Sign") (2000)
2. United States: Government paperwork elimination act ("GPEA") (2000)
3. European Parliament, Council: Directive 1999/93/ec of the european parliament and of the council of 13 December 1999 on a community framework for electronic signatures (2000)
4. Brasil: Medida provisoria no 2.200-2, de 24 de agosto de 2001 (2001)
5. ITU-T: Information technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks, 6th edn. Number ISO/IEC 9594-8 (December 2008)
6. Ellison, C., Schneier, B.: Ten risks of pki: What you're not being told about public key infrastructure. Computer Security Journal 16(1), 1–7 (2000)
7. Lopez, J., Oppliger, R., Pernul, G.: Why have public key infrastructures failed so far? Internet Research 15(5), 544–556 (2005)
8. Gutmann, P.: Pki: it's not dead, just resting. Computer 35(8), 41–49 (2002)
9. Rede Nacional de Ensino e Pesquisa (RNP): Infraestrutura de Chaves Públicas para Ensino e Pesquisa, ICPEdu (2013), `http://www.rnp.br/servicos/icpedu.html`
10. National Institute of Information Technology (ITI): Brazilian Public Key Infrastructure, ICP-Brazil (2013), `http://www.iti.gov.br/icp-brasil`
11. Merryman, J., Pérez-Perdomo, R.: The civil law tradition: an introduction to the legal systems of Europe and Latin America. Stanford University Press (2007)
12. Stallings, W.: Cryptography and Network Security: Principles and Practice, 5th edn. Prentice Hall (2010)

13. Lekkas, D., Gritzalis, D.: Cumulative notarization for long-term preservation of digital signatures. Computers Security 23(5), 413–424 (2004)
14. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 437–455. Springer, Heidelberg (1991)
15. Bayer, D., Haber, S., Stornetta, W.: Improving the efficiency and reliability of digital time-stamping. In: Sequences II: Methods in Communication, Security, and Computer Science, pp. 329–334 (1993)
16. ETSI: CMS Advanced Electronic Signatures (CAdES), 2.1.1 edn. Number TS 101 733 (March 2013)
17. ETSI: XML Advanced Electronic Signatures (XAdES), 1.4.2 edn. Number TS 101 903 (December 2010)
18. Custódio, R.F., Vigil, M.A.G., Romani, J., Pereira, F.C., da Silva Fraga, J.: Optimized certificates – A new proposal for efficient electronic document signature validation. In: Mjølsnes, S.F., Mauw, S., Katsikas, S.K. (eds.) EuroPKI 2008. LNCS, vol. 5057, pp. 49–59. Springer, Heidelberg (2008)
19. Vigil, M.A.G., Moecke, C.T., Custódio, R.F., Volkamer, M.: The notary based PKI – A lightweight PKI for long-term signatures on documents. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 85–97. Springer, Heidelberg (2013)
20. ETSI: Provision of harmonized Trust-service status information, 3.1.2 edn. Number TS 102 231 (December 2009)
21. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), Updated by RFC 5816 (August 2001)
22. Vigil, M.A.G., Cabarcas, D., Wiesmaier, A., Buchmann, J.: Authenticity, integrity and proof of existence for long-term archiving: a survey. Cryptology ePrint Archive, Report 2012/499 (2012), `http://eprint.iacr.org/`
23. Braun, J., Buchmann, J., Mullan, C., Wiesmaier, A.: Long term confidentiality: a survey. Cryptology ePrint Archive, Report 2012/449 (2012), `http://eprint.iacr.org/`

A Test-Bed for Intrusion Detection Systems Results Post-processing

Georgios Spathoulas[1], Sokratis K. Katsikas[2], and Anastasios Charoulis[3]

[1] University of Piraeus, Department of Digital Systems,
150 Androutsou St., Piraeus GR-18532, Greece
gspathoulas@unipi.gr
University of Central Greece, Department of Computer Science and Biomedical Informatics,
2-4 Papasiopoulou St., Lamia GR-35100, Greece
gspathoulas@ucg.gr

[2] University of Piraeus, Department of Digital Systems,
150 Androutsou St., Piraeus GR-18532, Greece
ska@unipi.gr

[3] University of Piraeus, Department of Digital Systems,
150 Androutsou St., Piraeus GR-18532, Greece
tcharoulis@unipi.gr

Abstract. Intrusion detection systems produce alert sets of low quality. Many post-processing methods have been proposed to make alert sets more meaningful to security analysts. Relevant research has to deal with an important task; implementing proposed methods and carrying out required experiments. In this paper we propose a platform which can be used as a test-bed for conducting intrusion detection alerts post-processing research. All the standard functionality is already implemented for the user, as she has to implement only the core logic of her method. Additionally the platform offer important reuse and evaluation capabilities. Finally we use the platform to implement a previous method of ours, in order to test its usefulness.

Keywords: intrusion detection, post-processing, platform, experiments.

1 Introduction

A lot of research has been focused on post-processing of intrusion detection alerts in recent years. Intrusion detection systems usually produce alert-sets of low quality. These alert-sets are characterized by their enormous size, which is disproportional to the size of the relevant protected systems, their high rate of false positives and false negatives and their inconsistency in regards to the real attack plan committed. Researches have been recently working on this field intensively and have proposed a lot of interesting methods that utilize ideas from various science fields such as machine learning, data mining, fuzzy logic,time series etc.

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 170–183, 2014.

These research efforts always include commonly used procedures such as reading alerts or evaluating results. Apart from that, researchers usually implement functionality they have already used in their previous work, while the most important problem is that comparison of experimental results is most of the time a big issue.

If segments of the methodologies proposed were formally defined as components, then they would be suitable for reuse. They could be reused by their authors or even by others. Apart from that, standard evaluation components could make the direct comparison of results easier and more elaborate.

This paper presents the development of a software platform, implemented in Java, that enables researchers to implement the post-processing solutions they have designed, as interconnected components. The platform contains well defined models of all the standard functionality needed by researchers and provides it to them as ready components. Additionally it offers to them a clear and easy way to inject their methods in the solution. Emphasis has been given on the reusability of solution's parts, in order for the user to be able to reuse her methods or distribute them to others. The evaluation part has also been standardized, in a way that results of different implementations are directly comparable.

In [12] an intrusion detection alerts post-processing filter is proposed to reduce false positives. The filter was developed in Java and consists of three algorithms, each one of which produces a validity score for each alert. Afterwards the scores produced for each alert are combined into one final score. The proposed platform could be used to re-implement this filter. Three components that would simulate the functioning of the algorithms along with one component that would be responsible for the fusion of the three scores could be created in the platform. The implementation effort needed would be significantly lower, while altering the final system or re-using parts of it would be very easy.

The remaining of the paper is structured as follows: In Section 2 we present and briefly discuss related work. Section 3 discusses the actual problems that exist for researchers trying to test their intrusion detection alerts post-processing methods. Section 4 presents the design concepts of our platform. Section 5 analyzes the implementation of our platform, while Section 6 presents the results we have obtained trying to use the platform to re-implement a previous method of ours.

2 Related Work

2.1 Post-processing Methods

In the past years many interesting alerts post processing solutions, have been proposed, incorporating methods from various science fields. The most important ones are analysed as follows:

In [14] authors discuss probabilistic alerts' similarity. They define a similarity calculation methodology. They try to aggregate alerts for which there is a relevant match into meta-alerts. For each new alert, they compute similarity to all existing meta-alerts. The alert is then merged with the best matching meta-alert, as long as their similarity passes a threshold value.

In [1] an analytical description of alerts' aggregation and correlation procedures is given. Authors propose an architecture that consists of multiple detection probes, the outputs of which are fed to aggregation and correlation components. In the aggregation phase the algorithm groups events together according to certain criteria. The aim is to discard multiple identical alerts in sensor level. In the correlation phase the algorithm creates correlation relationships between related events according to explicit rules. Once events are in a relationship, they are considered as part of the same attack and are processed together.

In [8] the motivation is to provide a framework for constructing attack scenarios through alert correlation, by using prerequisites and consequences of intrusions. The approach is based on the observation that alerts correspond to different stages of an attack scenario, with the earlier stages preparing for the later ones. Authors also developed an off-line tool on the basis of the formal framework, which tries to correlate alerts, by combining post conditions with prerequisites.

In [15] authors have implemented a complete system that tackles most aspects of alerts' post-processing and have conducted experiments on multiple data-sets to prove the validity of their assumptions. Their system consists of numerous components each responsible for a different task of post-processing. The alert-set is serially propagated through these components to the exit of the system. The system proposed is one of the most significant works in intrusion detection post-processing and has been influential for many other researchers.

2.2 Relative Platforms

Authors of previous subsection's work had to make important implementation efforts in order to test their systems. Our platform provides them with the tools to efficiently implement their post-processing methods. No similar platform, that tests intrusion detection alerts post-processing methods, has been proposed in literature. There are some systems presented in articles published in the early days of intrusion detection research, that focus on testing the intrusion detection systems themselves.

Authors in [10] and [9] propose a platform on which the user can create scripts that simulate intrusive or normal behaviour, in order to test an intrusion detection system. Then they systematically try to evaluate the performance of the intrusion detection system in test by observing the detected intrusions.

In [16] the system proposed injects dummy intrusion network traffic into the normal live traffic of a network. In this way a dataset for testing an intrusion detection system is created, while normal traffic is as realistic as possible. Moreover the user can test any intrusion she wishes, without creating any real security issues for the protected network.

In [7] signatures of Snort [11] network-based intrusion detection system are used as input to an event stream generator that produces randomized synthetic events that matches the input signatures. The resulting events stream is then used to trigger a number of different intrusion detection systems and the results are analyzed.

It is obvious that these efforts are related to testing the actual intrusion detection systems. Our motivation is to provide researchers with an elaborate alerts post processing methods development environment. There is no other system proposed in bibliography, that shares the same motivation.

3 The Problem

Intrusion detection has been a very intensively researched area in recent years. Many researchers work on this field and try to improve the quality of the results obtained by intrusion detection systems. While others try to achieve this by proposing improvements of the detection techniques, many researchers use post-processing of the produced alerts. They try to extract valuable knowledge of the security status of the system from the actual alert set.

Generally produced alert sets are of low quality. The most common problem is high false positives rate. The percentage of false alerts, is usually so high that it is hard to isolate the real alerts from false ones. Additionally the relevance between events and alerts is not always obvious. A single event may produce multiple identical instances of the same alert or it can produce many alerts that differ in small subset of their fields. Generally alerts are usually in lower level of complexity than the events that trigger them. All these factors contribute to the low quality of the produced alert set.

In general reading alert sets is impractical as they contain thousands of alerts, which are not all useful or they overlap, while many of them are false. The motivation of researchers dealing with alert post-processing is to improve the results obtained from the IDS in every possible way [6,3,17,13,2]. The main concepts in post-processing of intrusion detection alerts are :

- False positives reduction (filtering)
- Aggregation
- Correlation
- Clustering
- Visualization

Many researchers are working in the field of post processing of intrusion detection alerts, in order to enhance the produced alert-set. An important part of their efforts is dedicated to implementing their methods and justifying their performance with relative experiments. The main problems hindering these efforts are :

Researchers have to implement a lot of standard functionality irrelevant to their methods' logic. They have to write code for reading the alert-set from the intrusion detection system, transforming it to a format suitable for processing or measuring their method's performance in order to evaluate its efficiency.

Additionally if a researcher wants to extend or enhance a previous method of hers, then she has to recode all the functionality present in her previous implementation. While the functionality of the code will be similar, the researcher has to give a lot of attention to the changes needed to her existing code in order to function properly in the new implementation.

An important issue comes up when comparing methods of different authors. Their implementations vary along with the data they use or the evaluation methods they choose. This makes comparison problematic as different parameters in each implementation may induce doubts on the validity of the comparison itself. The reader finds it difficult to compare experimental results and conclude on which of the proposed methods is most suitable for a post-processing task.

The main principle on which the proposed system is based is to enable researchers to implement their methods in a more efficient and convenient way. The development effort required should be much lower while researchers should be given the opportunity to easily reuse or share their ideas.

4 Designing the System

The motivation behind this paper is to produce a platform, which will help the researchers on implementing their methods. The main ideas behind designing this platform were :

- Re-use of components implemented in previous methods
- Ready to use components for standard procedures (e.g. reading data from IDS)
- Included alert sets for widely used cases (e.g. alert set produced by Snort from DARPA [4],[5] datasets)
- Ready to use performance measuring components
- Ready to use visualization components

The system proposed enables researchers, in intrusion detection alerts post-processing field, to test the methods they propose in an efficient way. The main concept is that they should be able to develop components with one or more alert sets as inputs and one alert set as output. They should then use a graphical tool to connect these components (send ones output to another's input). The components along with the connections structure in which they are connected make up a solution in our platform. In this way researchers will be able to build sophisticated methods to improve the initial alert sets quality, with the minimum effort.

The main building blocks of the solution are the components. There are generic components that the user of the system may extend in order to achieve the post-processing functionality she has designed. There are also special components which are used to achieve specific functionality and the user is responsible for setting their parameters.

The flow of alert sets or in other words the connections between the components of the solution along with the details of each component are stored in an XML file. The XML file contains all the required information about each of the solution's components.

4.1 Abstract Solution Component

The user should implement the functionality she has designed. Our system provides her with all the infrastructure needed in order to start coding her logic. Every other aspect of the problem besides logic of the method such as reading data, sending data to other components, checking the validity of these data exchanges or measuring performance should be taken care of by our system. The researcher should be focused only on implementing her methods.

The abstract solution component is a Java Class that contains all the required characteristics that a component should have.

- Minimum number of inputs
- Maximum number of inputs
- List of accepted input types
- Output type
- Void execute() method that should be overridden by the Class implemented by the user

The user has to develop her own Java Class for

for each of the custom components used, the user has to develop her own Java Class which will extend the abstract solution component Class. The only requirement for the custom Classes is to override the execute() method of the abstract solution Class to implement the component's logic.

4.2 Special Components

There are special components that are used to achieve specific tasks needed for the experiments, such as reading data from an IDS source or measuring the performance of the system.

IDSDataReader : The IDSDataReader component is responsible for reading data (intrusion detection systems alerts) from a source and importing them into the system. This component is specific for each possible case of input. Input cases are characterized by two parameters; the IDS used and the format it keeps its data in. For example a IDSDataReader component can be developed to read data from a Snort installation that keeps data in a MYSQL database, while another would be needed to read data from a Snort installation that keeps alerts in a log file and a third one would be required to read data from a Bro IDS installation.

IDSEvaluator : The IDSEvaluator is another special component responsible for evaluating the performance of the solution proposed. The performance of the solution can be measured in various ways, e.g. how many false positives (alerts without a corresponding event) exist in the final alert set or how many false negatives exist (events without a corresponding alert). The evaluation is committed upon data that represent the real events that have taken place, while the alerts data-set was being collected. The format in which these data are fed to the IDSEvaluator component has to be predefined. An example is the XML format used by DARPA for the real events of DARPA data-sets.

4.3 Connecting the Components

After the researcher has implemented the required components, then the next step is to combine them, in order to produce a solution. The user uses a graphical interface to connect components, which is presented in Section 4.4. The system stores the produced solution in an XML file. This file contains information about each component such as :

- Id of the components
- Id's of previous components (their output is connected to the component's input)
- Id's of next components (the output of the component is connected to their inputs)
- Map containing values of configuration variables of the component

A subsection of a solution XML file that refers to a specific component is shown below :

```
<bean id=”id1” class=”component_class”>
    <property name=”previous”>
        <list>
            <ref bean=”id2”/>
        </list>
    </property>
    <property name=”next”>
        <list>
            <ref bean=”id3”/>
            <ref bean=”id4”/>
        </list>
    </property>
    <property name=”configMap”>
        <map>
            <entry key=”var1” value=”value1”/>
            <entry key=”var2” value=”value2”/>
        </map>
    </property>
</bean>
```

This entry for the component with id1 defines that its input comes from the output of the component with id2 and that its output is connected to the inputs of components with id3 and id4. This is depicted in Figure 1.

The functioning of the specific component is configured by the configMap property shown in the XML file that contains two configuration variables var1 and var2 along with the respective values.

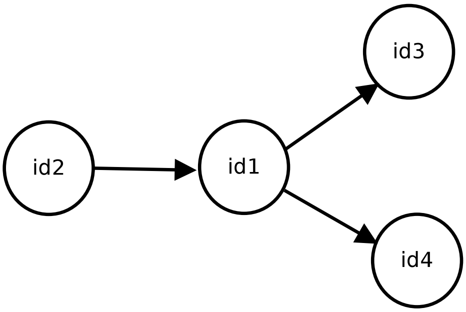

Fig. 1. The connections between components

4.4 Using the System

The proposed software contains a few initial components that implement methods we have developed in previous work of ours, which are enough for researchers to get the system going. Our aim is to create a public library of components, to which every researcher will be able to submit her components. If this library is sufficiently populated then :

- Everyone will have a lot of ready to use components to experiment with
- Developers of well performing components, will receive the analogous recognition
- Researchers will easily expand the work of others
- Comparison of the performance of different methods will be trivial

Of course every user is able to develop from scratch new components. As mentioned in section 4 the user has to define the acceptable range for the number of inputs, the accepted input types and the produced output type. The logic of the users method has to be implemented in a method execute() which should override an execute() method existing in the abstract component Class.

Then a graphical user interface, enables the user to create a structure of components. Through a drag and drop procedure the user can place components on the solution's canvas and then connect them in serial or parallel manner. The first component used has to be an IDataReader component, while the last should usually be an IDataEvaluator component. The structure created can be saved and loaded in the future. It can also be loaded in another installation of the platform as long as the required components exist in it. The user can export her components' Classes and import to them to another installation of the platform.

The solution (structure of components) created by the user is implemented as a directed graph of Java objects. Each component used is a node in this graph. When the user executes her solution the nodes of the graph are visited in a Breadth First Search (BFS) manner, beginning from the root of the graph. The execute() method of each component is run, while BFS algorithm's logic ensures that no component's execution is attempted, without first producing the required input.

5 Implementing the System

The platform has been developed in Java. In this section its main Classes are analyzed. They are presented in three subsections relevant to the components of the system, the data exchanged between these components and the user interface of the system.

5.1 Components Classes

The main Class of the system is the AbstractSolutionComponent Class. Developers that want to create their own components must write Classes that implement this Class. It contains all common functionality that components should have. AbstractSolutionComponnet Class main properties are :

- An id field, which is unique and representative of each component
- A set of Java Lists of Alert objects, that contain input alert sets
- A Java List of Alert objects, that contain output alert set
- Two Java Lists that contain previous and next components respectively and provide the means to create an interconnected diagram of components
- A boolean flag that shows if the component has been executed or not

The AbstractSolutioncomponent Class also has all the required methods, such as getters and setters for it's fields.

Every other implemented component's Class inherits AbstractSolutioncomponent Class . All the standard functions (input, output, etc) that a component should contain are implemented in AbstractSolutioncomponent Class. The only task that remains to the developer is override the execute() method and embed into it the core logic of her method.

The execute() method of each component should read input alert-sets from the inputAlertSet Lists, conduct the processing it has been designed to do and then store the resulting AlertSet to the outputAlertSet List. Before the execution of any other component which accepts this components output as input, the system will copy the outputAlertSet List of this component to the other components inputAlertSet Lists.

There is also an InitSolutionComponent Class, the objects of which are responsible for handling all the standard procedures for the solution. Reading data, validating solutions or evaluating results is committed by cooperating with other special Classes such as IDataReader, IdataEvaluator and InitializingComponent interface. All these Classes are depicted in Figure 2.

5.2 Data Classes

Data that flows between the components is mainly Java Lists of Alert Class objects. There is also an AlertId Class that relates alerts with their categories. Apart from that another Class used is TrueEvent that holds data of real events happened and is used by the IDataEvaluator Class. Figure 3 shows the relevant part of the Class diagram.

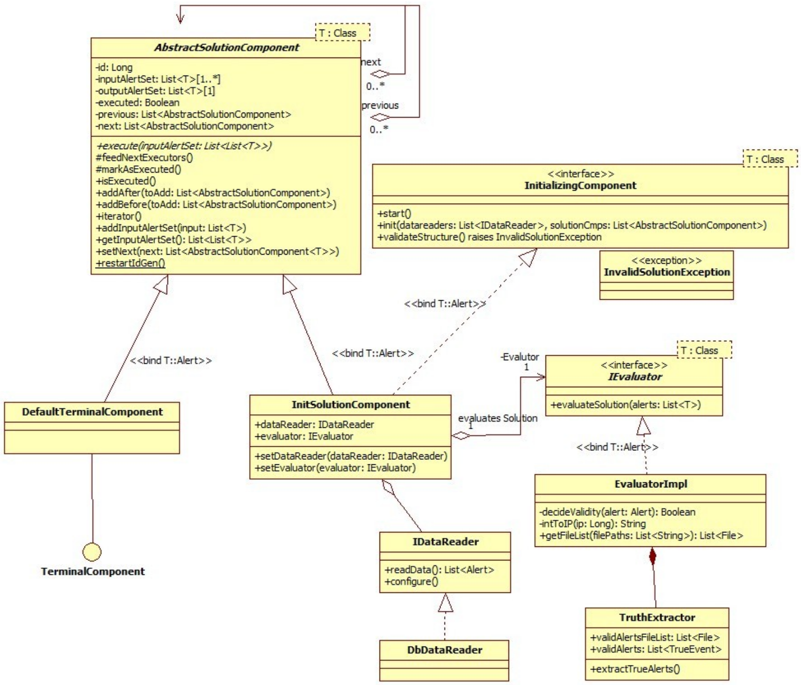
T : Class
AbstractSolutionComponent
-id: Long
-inputAlertSet: List<T>[1..*]
-outputAlertSet: List<T>[1]
-executed: Boolean
-previous: List<AbstractSolutionComponent>
-next: List<AbstractSolutionComponent>
+execute(inputAlertSet: List<List<T>>)
#feedNextExecutors()
#markAsExecuted()
+isExecuted()
+addAfter(toAdd: List<AbstractSolutionComponent>)
+addBefore(toAdd: List<AbstractSolutionComponent>)
+iterator()
+addInputAlertSet(input: List<T>)
+getInputAlertSet(): List<List<T>>
+setNext(next: List<AbstractSolutionComponent<T>>)
+restartIdGen()
next
0..*
previous
0..*
T : Class
<<interface>>
InitializingComponent
+start()
+init(datareaders: List<IDataReader>, solutionCmps: List<AbstractSolutionComponent>)
+validateStructure() raises InvalidSolutionException
<<exception>>
InvalidSolutionException
<<bind T::Alert>>
<<bind T::Alert>>
<<bind T::Alert>>
-Evalutor
1
T : Class
<<interface>>
IEvaluator
+evaluateSolution(alerts: List<T>)
DefaultTerminalComponent
InitSolutionComponent
+dataReader: IDataReader
+evaluator: IEvaluator
+setDataReader(dataReader: IDataReader)
+setEvaluator(evaluator: IEvaluator)
evaluates Solution
1
<<bind T::Alert>>
EvaluatorImpl
-decideValidity(alert: Alert): Boolean
-intToIP(ip: Long): String
+getFileList(filePaths: List<String>): List<File>
TerminalComponent
IDataReader
+readData(): List<Alert>
+configure()
DbDataReader
TruthExtractor
+validAlertsFileList: List<File>
+validAlerts: List<TrueEvent>
+extractTrueAlerts()

Fig. 2. Class Diagram of classes relevant components construction

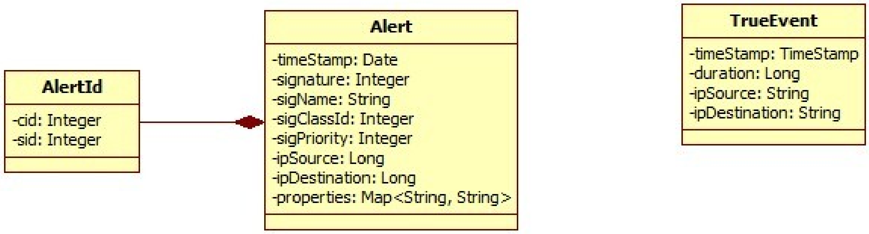

Fig. 3. Class Diagram of classes relevant to data types used in the platform

The Alert Class holds data relevant to alerts such as the time-stamp, the alert's signature, source IP, destination IP etc. There is a special field in the Alert Class, which holds a Java Map and is called properties. This can be utilized by the user, in order to enhance the basic data type (simple alert) with meta-data. For example a component can calculate a validity score for each alert. This score can be attached to the alert itself, by including it in the properties map. The next component that will accept the enhanced alert set as input will be able to read and utilize this validity score.

AlertId Class relates alerts to their Classes by defining alert id to Class id relationships. This may be used by a component that needs attack class information for its processing.

The TrueEvent Class holds data relevant to the true security events occurred. Its objects contain information such as the time stamp, the duration and source and destination IPs. This is used by the IDataEvaluator Class, in order to check if the alerts of the finally produced alert-set are valid or not. If events, relevant to an alert, exist in the List of TrueEvent objects then this alert is marked as valid, otherwise it is marked as false.

5.3 User Interface Classes

Finally the third part of Classes of our platform enables the user to create her solution with a intuitive graphical interface.

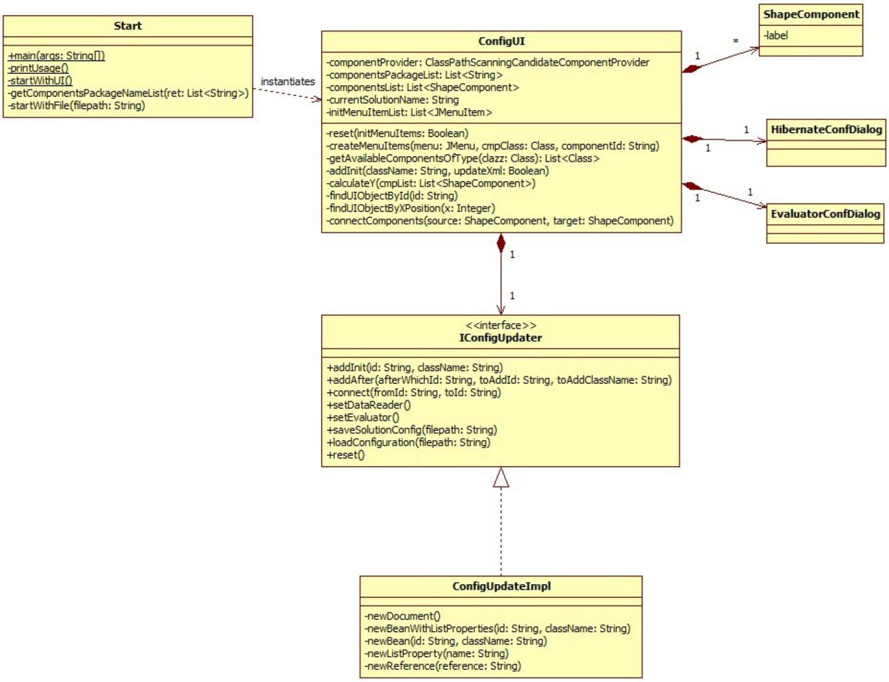
Start
+main(args: String[])
-printUsage()
-startWithUI()
-getComponentsPackageNameList(ret: List<String>)
-startWithFile(filepath: String)
instantiates
ConfigUI
-componentProvider: ClassPathScanningCandidateComponentProvider
-componentsPackageList: List<String>
-componentsList: List<ShapeComponent>
-currentSolutionName: String
-initMenuItemList: List<JMenuItem>
-reset(initMenuItems: Boolean)
-createMenuItems(menu: JMenu, cmpClass: Class, componentId: String)
-getAvailableComponentsOfType(clazz: Class): List<Class>
-addInit(className: String, updateXml: Boolean)
-calculateY(cmpList: List<ShapeComponent>)
-findUIObjectById(id: String)
-findUIObjectByXPosition(x: Integer)
-connectComponents(source: ShapeComponent, target: ShapeComponent)
1
*
ShapeComponent
-label
1
1
HibernateConfDialog
1
1
EvaluatorConfDialog
1
1
<<interface>>
IConfigUpdater
+addInit(id: String, className: String)
+addAfter(afterWhichId: String, toAddId: String, toAddClassName: String)
+connect(fromId: String, toId: String)
+setDataReader()
+setEvaluator()
+saveSolutionConfig(filepath: String)
+loadConfiguration(filepath: String)
+reset()
ConfigUpdateImpl
-newDocument()
-newBeanWithListProperties(id: String, className: String)
-newBean(id: String, className: String)
-newListProperty(name: String)
-newReference(reference: String)

Fig. 4. Class Diagram of classes relevant to user interface

The main Class in this part of the platform is ConfigUI. This Class holds all the information required for the graphical representation of the solution. Its main properties are a list of all the components of the solution, a list of all the required packages in an installation for the solution to work and the name of the solution. All the required methods that enable the graphical user interface to function, also exist in this Class.

6 Testing the System

In order to test the system we have used it to re-implement a previous work of ours. Then we compared the effort needed in both cases and highlighted the qualitative advantages of using the proposed system.

In [12] we proposed a post-processing filter, to reduce false positives in network-based intrusion detection systems. The filter consists of three components, the functioning of which is based upon statistical properties of the input alert set. The filter shown in Figure 5 was developed in Java for the purposes of the relevant experiments that justified that it is able to drastically filter out false alerts. There are three components; namely NRA, HAF and UFP. Each one produces a score for each alert, which indicates the probability of this alert to be true. Afterwards these three scores produced for each alert are combined into one final score. A threshold is finally used to identify the alerts that will be rejected.

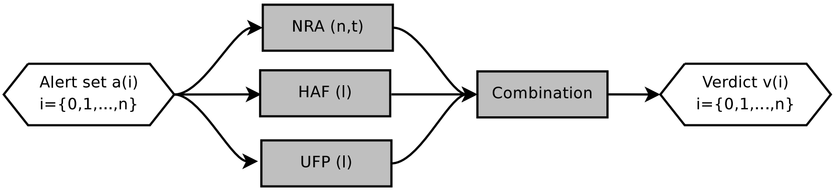

Fig. 5. The filter

The same filter has been developed by using the proposed system. We have taken advantage of ready to use functionality such as reading alerts from Snort and measuring the performance of the filter, as this functionality is implemented by standard components of the system. We developed three components one for each component of the original filter by inheriting the AbstractSolutionComponent Class. Additionally a component responsible for the fusion of the results obtained by each component have been created.

As it was expected implementing the filter with the proposed system demanded marginally less lines of code and less effort from the developer. It has been calculated that the lines of code needed to be written in the scenario of using the system were approximately 40% of the lines of code in the original implementation of the filter. This mainly happened because the original implementation was characterized by a lot of code redundancy. Parts of code, irrelevant to filters logic, appears multiple times throughout the initial implementation. This code handled standard procedural functioning such as receiving an alert-set, exporting an alert-set from a component, calculating false positives rate etc. This functioning is already implemented in our system and the developer can focus on writing code only for the logic of her method.

Moreover each component developed in our system is built independently of the rest of the solution. This means that it can be easily moved out of the solution, edited, replaced by another or even distributed to others. So if this method or part of it is needed to be used in a future work of ours, the procedure of re-using it will be trivial.

Testing the platform has indicated that in practice it can be of great use to researchers. It is capable of significantly reducing the effort they have to put in to implement their methods. It also enables them to efficiently support their ongoing research by easily expanding previous methods of theirs. Finally it provides cooperation capabilities, as researches can effortlessly exchange their methods in the format of ready-to-use components.

7 Discussion and Future Work

It is generally accepted that post-processing of alerts is a significant area of intrusion detection research. All the authors proposing a relevant method have to put a lot of effort on implementation, in order to prove their method's validity. The implementation part is always difficult and time consuming. There are no tools, that can help researchers on this problem, so they have to manually code everything.

Our platform, presented in this paper, fills this gap. We offer to researchers ready to use functionality, the ability to reuse theirs or others older functionality and a standard evaluation environment that enables reliable comparisons between different methods. Our test has demonstrated that using our platform makes the implementation easier and less time-consuming.

Our platform is in its initial steps and future work can add value to its use from researchers. A lot of attention has to be given on making users publish their components. The true power of our platform is that it enables easy reuse of previous methods. The researcher has to just import others' components to her installation to make use of them, so testing or extending others' work is very easy. So if researching community made their components available for public use, then the community itself would benefit from an important repository of ready to use components.

Apart from that, obliging users to write in Java is a limitation that should be vanished. If a researcher has already implemented his methods in another programming language (C or Matlab), she will not re-implement it in Java just to make it public to others. We should implement a generic component in our system that will be able to communicate with external software (functionality implemented in other languages) and use it in terms of the solution designed in our platform.

Finally, in order to accommodate functionality, that demands excessive processing power, we should implement a second abstract component that will be designed in parallel programming approach. A relevant hardware platform should be chosen and a component that will execute different parts of its execute() method on different processors available should be created. In this way the user that has in her disposal the required hardware, will be able to exploit it in order to implement and test a complicated, processing power demanding method.

References

1. Debar, H., Wespi, A.: Aggregation and correlation of intrusion-detection alerts. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 85–103. Springer, Heidelberg (2001)
2. Dumas, M., Robert, J.-M., McGuffin, M.J.: Alertwheel: radial bipartite graph visualization applied to intrusion detection system alerts. IEEE Network 26(6), 12–18 (2012)
3. Hubballi, N., Biswas, S., Nandi, S.: Network specific false alarm reduction in intrusion detection system. Security and Communication Networks 4(11), 1339–1349 (2011)
4. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 darpa offline intrusion detection evaluation. Comput. Netw. 34(4), 579–595 (2000)
5. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX 2000, vol. 2, pp. 12–26 (2000)
6. Maggi, F., Matteucci, M., Zanero, S.: Reducing false positives in anomaly detectors through fuzzy alert aggregation. Information Fusion 10(4), 300–311 (2009)
7. Mutz, D., Vigna, G., Kemmerer, R.: An experience developing an ids stimulator for the black-box testing of network intrusion detection systems. In: Proceedings of the 19th Annual Computer Security Applications Conference, pp. 374–383 (December 2003)
8. Ning, P., Cui, Y., Reeves, D.S.: Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 245–254. ACM (2002)
9. Puketza, N., Chung, M., Olsson, R.A., Mukherjee, B.: A software platform for testing intrusion detection systems. IEEE Software 14(5), 43–51 (1997)
10. Puketza, N.J., Zhang, K., Chung, M., Mukherjee, B., Olsson, R.A.: A methodology for testing intrusion detection systems. IEEE Transactions on Software Engineering 22(10), 719–729 (1996)
11. Roesch, M.: Snort - Lightweight Intrusion Detection for Networks. In: System Administration Conference (1999)
12. Spathoulas, G.P., Katsikas, S.K.: Reducing false positives in intrusion detection systems. Computers & Security 29(1), 35–44 (2010)
13. Thomas, C., Balakrishnan, N.: Improvement in intrusion detection with advances in sensor fusion. IEEE Transactions on Information Forensics and Security 4(3), 542–551 (2009)
14. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
15. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Transactions on Dependable and Secure Computing 1, 146–169 (2004)
16. Wan, T., Yang, X.D.: Intrudetector: a software platform for testing network intrusion detection algorithms. In: Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC 2001, pp. 3–11 (December 2001)
17. Zhou, C.V., Leckie, C., Karunasekera, S.: Decentralized multi-dimensional alert correlation for collaborative intrusion detection. Journal of Network and Computer Applications 32(5), 1106–1123 (2009)

Uncertainty in Intrusion Detection Signaling Games

Ioanna Kantzavelou[1] and Sokratis K. Katsikas[2]

[1] Dept. of Informatics, Technological Educational Institute of Athens,
Ag. Spiridonos St., 122 10 Aigaleo, Athens, Greece
ikantz@teiath.gr

[2] Dept. of Digital Systems, University of Piraeus,
150 Androutsou St., 185 32 Pireaus, Greece
ska@unipi.gr

Abstract. In the area of Intrusion Detection (ID) games are being played between potential attackers and the Intrusion Detection Systems (IDSs) that protect the target systems of several attacks. More than a few game models presented in the past have showed how much beneficial Game Theory could be when incorporating with ID. In this research work an ID game model is constructed and examined as a signaling game. First, we construct the ID signaling game in an extensive form by defining the corresponding payoffs. Next, we represent it in a payoff matrix as a normal form game. We examine then the solution of the game by removing the dominated strategies. Finally, we compute all the equilibria of the ID signaling game in pure and behavioral strategies. The results give valuable explanations about how ID games are being played, what are players' choices and under which circumstances, and the amount of uncertainty an ID game bears from its start point until the end.

Keywords: intrusion detection, insiders, game theory, signaling game.

1 Introduction

Attempting to model ID as a signaling game, we assume a User who has already gained access to a Target System (TS) and starts using it, regardless whether he is a legitimate user, a masquerader, a hacker, or a cracker. The method used to get into the system, in cases where the User is not authorized, are not considered here because it is covered by the area of Access Control in IT Security. In our case, we care about users of the system we do not know whether they are going to behave legitimately or illegally, accidentally or intentionally.

In signaling games players have no complete information and therefore they exchange signals to play the game. A signal reflects private information a player holds and its recipient encodes it in order to take an action.

Osborne and Rubinstein describe in [5] signaling games as Bayesian extensive games with observable actions in a simple form. Likewise, we formulate, in the next section, the interactions between a user of the system (normal or attacker) and the IDS, as a signaling game.

S. Katsikas and I. Agudo (Eds.): EuroPKI 2013, LNCS 8341, pp. 184–203, 2014.

Assuming an Intrusion Detection System installed on a target system as a second line of defense, there is a user of this system whose behavior is double named, usually he acts normally, but, occasionally he acts illegally by breaching the security policy of the system. Each time the user is acting, he sends a signal, which is received by the IDS. The IDS decodes this signal as an attempt to distinguish the action between normal and attacking. These interactions form a signaling game with two players, the "sender" (*User*) and the "receiver" (*IDS*). The *User* is informed of the value of an uncertain parameter θ_1, which is his ability to commit a specific type of attack, while the *IDS* does not know it. Similarly, the *IDS* has been designed to detect these types of attacks with an expected detection rate of value θ_1.

Then the *User* chooses an action, which will be received by the *IDS*, as a message m is being transmitted from a sender to a receiver. The *IDS* observes the action, but it cannot observe the value of the uncertain parameter θ_1. Its IDS's turn to take an action a. There are three factors each player's payoff depends upon; the value of parameter θ_1, the message m sent by the sender, and the action a taken by the receiver [5].

We need to know how ID signaling games are being played, what players choose to play, what players' know when they choose an action, how much certain they are concerning their choices and their beliefs of what the opponent knows, in order to give directions and control such a game for the benefit of the IDS and the TS.

The paper has been organized in sections. The construction of the proposed signaling game model is illustrated in Section 2. The normal form of the ID signaling game that models the interactions between players is constructed in Section 3 and solved in Sections 4 and 5. We briefly review related works in Sec. 6. Finally, in Section 7, we summarize our research work by evaluating the model and its operation, and we give suggestions for future research directions.

2 Constructing the ID Signaling Game

The *User* will move in two ways, acting *Legitimately* (L) or acting *Illegally* (I). The Target System is equipped with an Intrusion Detection System (IDS) ready to play with this User. The *IDS* will decide to *Prevent* (P) the User further using the TS, or to allow the User to *Continue* (C) working with it. To make this decision, the IDS should conclude if the User is an enemy of the TS, i.e. an Attacker, regardless he is an insider or an outsider who has stolen the identity of a internal user of the system (masquerader) and gained access.

The IDS does not know for sure if the *User* is a *Normal User* or an *Attacker*. This means that there is a simple probability distribution. Assuming that the number of reported attacks is for example the 25% of the occurring events in a Target System, then the *IDS* knows with probability $\frac{1}{4}$ that the *User* is an *Attacker* and with probability $\frac{3}{4}$ that the *User* is a *Normal User*. Later on, this number will be refined to reflect the actual number of attacks that take place in this specific Target System. This means that the proposed system will be self tuning and adjustable to current data related to the Target System itself.

Examining the set of alternative circumstances, the *IDS* will prevent the *User* if he is an *Attacker* and the *Attacker* will run off, because he was caught by the *IDS*. But if the IDS prevents a *Normal User* from using the Target System, then this *Normal User* might request justice, because a false positive alarm has been raised against him unfairly. The *IDS* receives signals from the User, and the decision whether he is a *Normal User* or an *Attacker* derives from the examination of these signals.

2.1 Defining the Payoffs

The set of actions for every player is a compact subset of the Euclidean space $\mathbb{R}^2$, and because it is finite, the game is a finite game. The set of profiles, corresponding to the set of pure actions, is the combination of actions, one action for each player, defined as the Cartesian product,

$$A \equiv \prod_{i \epsilon N} A_i .$$

In the ID signaling game, the set of pure action profiles is defined by

$$A \equiv A_{User} \times A_{IDS} = \{(L, C), (I, C), (L, P), (I, P)\}.$$

The number of profiles that are elements of this set is $k \times m = 2 \times 2 = 4$. We indicate a member of a profile with $(x_i)_{i \epsilon N}$ or simply (x_i), where N is the set of players $N = \{User, IDS\}$.

Next we specify players' preference rankings over the action profiles. Each player $i \epsilon N$ ranges the action profiles from the most preferred to the least one. A preference relation $\succeq_i$ on the set $A = x_{i \epsilon N} A_i$ for player i specifies a binary relation, represented by a payoff function $u_i : A \rightarrow \mathbb{R}$. The function u is a continuous function, known also as von Neumann-Morgenstern utility function. For two pure actions a_{i_1} and a_{i_2} of player i, $u_i(a_{i_1}) \geq u_i(a_{i_2})$, whenever $a_{i_1} \succeq a_{i_2}$. The values of this function are called payoffs or utilities [5].

The utility functions U_N for normal user and U_A for attacker determine the corresponding payoffs at each node of the extensive form game and at each cell of the normal form game respectively. As for the utility function U_{IDS}, we adjust it to reflect all the preferences, when the opponent is either a *Normal User* or an *Attacker* in a signaling game, as described in the sequel.

Because when playing with different users, the user type matters, we consider the preferences of two different types of user players, a normal user and an attacker, and the preferences of the IDS when playing with different user players. Then, we construct the corresponding utility functions following Binmore's method [1], to quantify the outcomes of the proposed game in a variety of instances.

Normal User's Preferences. Interpreting the action profiles when the user player is a normal user of a system, we consider the corresponding set of a Normal User's preferences, denoted by $\mathcal{N}$. This set includes the following four items:

$$\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4\}$$

where,

$\mathcal{N}_1$: A Normal User is acting legitimately and the IDS allows him to continue.
$\mathcal{N}_2$: A Normal User is being prevented by the IDS although he is acting legitimately.
$\mathcal{N}_3$: A Normal User is acting illegally but the IDS allows him to continue.
$\mathcal{N}_4$: A Normal User is being prevented by the IDS because he is acting illegally.

For a Normal User it is most desirable to act legitimately without preventions and his next choice is to act illegally with no stops, because illegal actions are not intentional. Similarly, he prefers the IDS to prevent his actions when these are illegal rather than legitimate. Based on these lines of reasoning, the ranking of these preferences from the less preferred to the most one gives the following:

$$\mathcal{N}_3 \prec \mathcal{N}_1 \text{ and } \mathcal{N}_2 \prec \mathcal{N}_4.$$

In order to get Normal User's preferences fully ordered, there is a need for connection between these two relations. A Normal User most prefers $\mathcal{N}_1$ and his worst choice is $\mathcal{N}_2$. Additionally, examining his preferences between $\mathcal{N}_3$ and $\mathcal{N}_4$, a Normal User prefers $\mathcal{N}_3$ because he has no intention to harm the TS, so he wants an uninterruptible use of it. This interpretation results into the following chain of preferences:

$$\mathcal{N}_2 \prec \mathcal{N}_4 \prec \mathcal{N}_3 \prec \mathcal{N}_1 \tag{1}$$

From this set of preferences and the defined relations that reflect the ranking, we will define the corresponding utility function for the Normal User. Suppose that $U_{\mathcal{N}} : \{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4\} \rightarrow \mathbb{R}$ is the utility function of the Normal User. With regard to his preferences, the worst action profile is $\mathcal{N}_2$. So, $U_{\mathcal{N}}(\mathcal{N}_2) = 0$. At the other end, he mostly prefers $\mathcal{N}_1$. Therefore, $U_{\mathcal{N}}(\mathcal{N}_1) = 1$. Selecting anyone between the other two preferences in the middle of the rank, we assign $\frac{1}{2}$ utility to the preference $\mathcal{N}_4$, that is, $U_{\mathcal{N}}(\mathcal{N}_4) = \frac{1}{2}$. Finally, because $\mathcal{N}_3$ is the intermediate between $\mathcal{N}_4$ and $\mathcal{N}_1$, we define $U_{\mathcal{N}}(\mathcal{N}_3) = \frac{3}{4}$, by dividing the distance between $U_{\mathcal{N}}(\mathcal{N}_1)$ and $U_{\mathcal{N}}(\mathcal{N}_4)$ with 2.

In this way, instead of having a ranking of encoded preferences as presented in expression (1), we have real numbers to represent preference relations, very handy for calculations. This is a more convenient representation when making choices, where the criterion is the maximization of the utility function $U_{\mathcal{N}}$. Table 1 below summarizes in the second row the specified utilities for the Normal User. The third row describes the corresponding utilities free of fractions, after multiplying them by 4.

Table 1. Normal User's Utility Function

x	$\mathcal{N}_2$	$\mathcal{N}_4$	$\mathcal{N}_3$	$\mathcal{N}_1$
$U_{\mathcal{N}}(x)$	0	$\frac{1}{2}$	$\frac{3}{4}$	1
$4 \cdot U_{\mathcal{N}}(x)$	0	2	3	4

Attacker's Preferences When the user player is an attacker, we consider in a similar way the set of an Attacker's preferences, denoted by $\mathcal{A}$.

$$\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\},$$

where,

$\mathcal{A}_1$: An Attacker does not achieve his goals and he is not being detected.
$\mathcal{A}_2$: An Attacker does not achieve his goals and he is being detected and stopped by the IDS.
$\mathcal{A}_3$: An Attacker achieves his goals without being detected.
$\mathcal{A}_4$: An Attacker achieves his goals and is being detected and stopped by the IDS.

As for an Attacker, the most preferable outcomes of the game might be those where he is achieving his goals. Between being detected or not, he prefers the second. In addition, he does not prefer to be prevented when acting legitimately, but he prefers to continue. Ranking these preferences from the less preferred to the most one, we get:

$$\mathcal{A}_2 \prec \mathcal{A}_1 \text{ and } \mathcal{A}_4 \prec \mathcal{A}_3$$

To connect the above relations and find an ordered ranking of Attacker's preferences, we examine further his profile. Taking into account that because he mostly prefers to achieve his goals no matter whether he will be detected or not, he is dedicated to his goals. Therefore, the other two preferences will eventually follow. This explanation results into the following ordered attacker's preferences:

$$\mathcal{A}_2 \prec \mathcal{A}_1 \prec \mathcal{A}_4 \prec \mathcal{A}_3 \qquad (2)$$

Similarly, we will define the corresponding utility function for the Attacker, based on the set of preferences $\mathcal{A}$ and the defined preference relations. Suppose that $U_{\mathcal{A}} : \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\} \to \mathbb{R}$ is the utility function of the Attacker. With regard to his preferences, an Attacker dislikes action profile $\mathcal{A}_2$ and prefers mostly $\mathcal{A}_3$. For this reason, we define $U_{\mathcal{A}}(\mathcal{A}_2) = 0$ and $U_{\mathcal{A}}(\mathcal{A}_3) = 1$, respectively. If we select $\mathcal{A}_1$ as one intermediate between $\mathcal{A}_1$ and $\mathcal{A}_4$ which are left, we define its utility as $U_{\mathcal{A}}(\mathcal{A}_1) = \frac{1}{2}$. Finally, we calculate $U_{\mathcal{A}}(\mathcal{A}_4)$, following the same reasoning as we did before for the Normal User, and we define $U_{\mathcal{A}}(\mathcal{A}_4) = \frac{3}{4}$. In Table 2

Table 2. Attacker's Utility Function

x	$\mathcal{A}_2$	$\mathcal{A}_1$	$\mathcal{A}_4$	$\mathcal{A}_3$
$U_{\mathcal{A}}(x)$	0	$\frac{1}{2}$	$\frac{3}{4}$	1
$4 \cdot U_{\mathcal{A}}(x)$	0	2	3	4

below, the second row summarizes the defined utilities for the Attacker. The third row describes the corresponding utilities free of fractions, after multiplying them by 4.

In a case where an Attacker has another profile, the preference ranking would be totally different. For example, if an Attacker is an internal attacker, an insider of the Target System, then he mostly prefers not to be detected rather than attacking. His preferences derive from the double role he plays, the mixture between a Normal user and an Attacker too. In such a situation, the ranking of his preferences might be as below:

$$\mathcal{A}_2 \prec \mathcal{A}_4 \prec \mathcal{A}_1 \prec \mathcal{A}_3 \tag{3}$$

IDS's Preferences. The set of an IDS's preferences is denoted by $\mathcal{IDS}$ and includes eight items in a signaling game, as described in the sequel:

$$\mathcal{IDS} = \{\mathcal{IDS}_1, \mathcal{IDS}_2, \mathcal{IDS}_3, \mathcal{IDS}_4, \mathcal{IDS}_5, \mathcal{IDS}_6, \mathcal{IDS}_7, \mathcal{IDS}_8\},$$

where,

$\mathcal{IDS}_1$: The IDS allows a Normal User who is acting legitimately to continue.
$\mathcal{IDS}_2$: The IDS prevents a Normal User who is acting legitimately to continue.
$\mathcal{IDS}_3$: The IDS allows a Normal User who is acting illegally to continue.
$\mathcal{IDS}_4$: The IDS prevents a Normal User who is acting illegally to continue.
$\mathcal{IDS}_5$: The IDS allows an Attacker who is acting legitimately to continue.
$\mathcal{IDS}_6$: The IDS prevents an Attacker who is acting legitimately to continue.
$\mathcal{IDS}_7$: The IDS allows an Attacker who is acting illegally to continue.
$\mathcal{IDS}_8$: The IDS prevents an Attacker who is acting illegally to continue.

Ranking these preferences from the most disliked to the most preferred one, we get:

$$\mathcal{IDS}_7 \prec \mathcal{IDS}_5 \prec \mathcal{IDS}_2 \prec \mathcal{IDS}_3 \prec \mathcal{IDS}_1 \prec \mathcal{IDS}_4 \prec \mathcal{IDS}_6 \prec \mathcal{IDS}_8 \tag{4}$$

Next, we will define another utility function for the player IDS. Suppose that $U_{\mathcal{IDS}} : \{\mathcal{IDS}_1, \mathcal{IDS}_2, \mathcal{IDS}_3, \mathcal{IDS}_4, \mathcal{IDS}_5, \mathcal{IDS}_6, \mathcal{IDS}_7, \mathcal{IDS}_8\} \to \mathbb{R}$ is the utility function for the IDS. The IDS has an aversion to preference $\mathcal{IDS}_7$, because this is the worst case scenario for it that raises a false negative alarm. For this reason, we define $U_{\mathcal{IDS}}(\mathcal{IDS}_7) = 0$. Furthermore, because it mostly prefers $\mathcal{IDS}_8$, we define $U_{\mathcal{IDS}}(\mathcal{IDS}_8) = 1$. Selecting between $\mathcal{IDS}_3$ and $\mathcal{IDS}_1$, which are intermediate preferences, we decide to define $U_{\mathcal{IDS}}(\mathcal{IDS}_3) = \frac{1}{2}$. Next, because $\mathcal{IDS}_4$ is the intermediate between $\mathcal{IDS}_3$ and $\mathcal{IDS}_8$, we define $U_{\mathcal{IDS}}(\mathcal{IDS}_4) = \frac{3}{4}$, by calculating the value of $U_{\mathcal{IDS}}(\mathcal{IDS}_4)$ which is the middle between $\mathcal{IDS}_3$ and $\mathcal{IDS}_8$, that is:

$$U_{\mathcal{IDS}}(\mathcal{IDS}_4) = U_{\mathcal{IDS}}(\mathcal{IDS}_3) + \frac{U_{\mathcal{IDS}}(\mathcal{IDS}_8) - U_{\mathcal{IDS}}(\mathcal{IDS}_3)}{2} = \frac{1}{2} + \frac{1-\frac{1}{2}}{2} = \frac{1}{2} + \frac{\frac{1}{2}}{2} = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}.$$

Calculating the utilities for $\mathcal{IDS}_1$ and $\mathcal{IDS}_6$ respectively, we get:

$$U_{\mathcal{IDS}}(\mathcal{IDS}_1) = U_{\mathcal{IDS}}(\mathcal{IDS}_3) + \frac{U_{\mathcal{IDS}}(\mathcal{IDS}_4) - U_{\mathcal{IDS}}(\mathcal{IDS}_3)}{2} = \frac{1}{2} + \frac{\frac{3}{4}-\frac{1}{2}}{2} = \frac{1}{2} + \frac{\frac{1}{4}}{2} = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}.$$

$$U_{\mathcal{IDS}}(\mathcal{IDS}_6) = U_{\mathcal{IDS}}(\mathcal{IDS}_4) + \frac{U_{\mathcal{IDS}}(\mathcal{IDS}_8) - U_{\mathcal{IDS}}(\mathcal{IDS}_4)}{2} = \frac{3}{4} + \frac{1-\frac{3}{4}}{2} = \frac{3}{4} + \frac{\frac{1}{4}}{2} = \frac{3}{4} + \frac{1}{8} = \frac{7}{8}.$$

Finally, we calculate in a similar way the utilities for $\mathcal{IDS}_5$ and $\mathcal{IDS}_2$ as described in the sequence:

$$U_{\mathcal{IDS}}(\mathcal{IDS}_5) = U_{\mathcal{IDS}}(\mathcal{IDS}_7) + \frac{U_{\mathcal{IDS}}(\mathcal{IDS}_3) - U_{\mathcal{IDS}}(\mathcal{IDS}_7)}{3} = 0 + \frac{\frac{1}{2}-0}{3} = \frac{\frac{1}{2}}{3} = \frac{1}{6}.$$

$$U_{\mathcal{IDS}}(\mathcal{IDS}_2) = U_{\mathcal{IDS}}(\mathcal{IDS}_5) + \frac{U_{\mathcal{IDS}}(\mathcal{IDS}_3) - U_{\mathcal{IDS}}(\mathcal{IDS}_5)}{2} = \frac{1}{6} + \frac{\frac{1}{2}-\frac{1}{6}}{2} = \frac{1}{6} + \frac{\frac{2}{6}}{2} = \frac{1}{6} + \frac{1}{6} = \frac{2}{6} = \frac{1}{3}.$$

In Table 3, we summarize the utilities defined for the IDS, in the second row. The third row contains the corresponding utilities transformed into integer numbers instead of fractions.

Table 3. IDS's Utility Function in a Signaling Game

x	$\mathcal{IDS}_7$	$\mathcal{IDS}_5$	$\mathcal{IDS}_2$	$\mathcal{IDS}_3$	$\mathcal{IDS}_1$	$\mathcal{IDS}_4$	$\mathcal{IDS}_6$	$\mathcal{IDS}_8$
$U_{\mathcal{IDS}}(x)$	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{5}{8}$	$\frac{3}{4}$	$\frac{7}{8}$	1
$24 \times U_{\mathcal{IDS}}(x)$	0	4	8	12	15	18	21	24

Regarding the payoffs of this game on behalf of the *User*, if the *IDS* permits a *Normal User* to *Continue*, then the *Normal User* gains 4 points when he is acting *Legitimately*, and 1 point less, i.e. he gets 3 points when he is acting *Illegally.* Moreover, if the *IDS* permits an *Attacker* to *Continue*, then the *Attacker* gains 2 points if he acts *Legitimately* and he doesn't achieve his goals, and 4 points if he acts *Illegally* and he achieves his goals.

Similarly, if the *IDS Prevents* a *Normal User* to use the Target System, then the *Normal User* gets no points (0 points) if he acts *Legitimately* and he gets 2 points if he acts *Illegally*, because he does not act by purpose. Likewise, if the *IDS Prevents* an *Attacker* to use the Target System, then the *Attacker* gets nothing if he acts *Legitimately* and 3 points if he acts *Illegally.*

As the *User*'s payoffs start from 0 and goes to 4, the *IDS*'s payoffs vary between 0 and 24. There is a difference between the two payoffs' scales, because a *Normal User* has 4 payoffs, an *Attacker* another 4, whereas the *IDS* has 8 payoffs, since he might play the game with any of them, either the *Normal User* or the *Attacker.*

Specifically, the *IDS* gains 15 points if it permits a *Normal User* who acts *Legitimately* to *Continue*, and 12 points if it permits a *Normal User* to *Continue* although he acts *Illegally.* In the case it permits an *Attacker* to *Continue* because he acts *Legitimately*, the *IDS* gains only 4 points, because, this is a false negative alarm. If it *Prevents* a *Normal User* to *Continue* although he acts *Legitimately*, the *IDS* gets 8.

In addition, the *IDS* loses by getting no points at all, when it permits an *Attacker* with *Illegal* actions to *Continue.* On the contrary, the *IDS* gains 18 points if it *Prevents* a *Normal User* from acting *Illegally*, 21 points if it *Prevents* an *Attacker* from acting *Legitimately*, and finally, 24 points if it *Prevents* an *Attacker* from acting *Illegally.*

Apparently, the ID game as a signaling game is not a zero-sum game, neither a constant-sum game. An *Attacker* is pretty happy if he commits an attack without being caught by the *IDS* (4 points), but he is a loser if the *IDS* detects his intentions correctly and stops him before he achieves his goals (0 points).

In the same way, a *Normal User* is satisfied by using the Target System in a *Legitimate* manner and nobody disturbs or stops him. But when the *IDS Prevents* him unfairly from doing so, he is one hundred per cent a loser of the game.

Conversely, the *IDS* maximum payoff is when it detects accurately an *Attacker* who acts *Illegally* and stops him (24 points), that is, when the *User* also gets some payoff (3 points), because he has already acted *Illegally.* Finally, the *IDS* gets no payoff (0 points) when it leaves undetected an *Attacker* who acts *Illegally* and permits him to *Continue* using the TS.

Figure 1 shows the extensive form of the ID game as a signaling game, drawn by the GAMBIT tool [2].

Since the ID game starts with a *Chance* node, where there is a probability p with which a *User* is a *Normal User*, and a probability $1 - p$ with which the *User* is an *Attacker*, the game is an incomplete information game, because the *IDS* does not know for sure what is the type of *User* it is interacting (playing) with. This is the private information the *User* holds.

Incomplete information games were formulated and studied by John Harshanyi in 1967[1]. They are amongst the most challenging games to be solved.

[1] Harshanyi shares the 1994 Nobel Prize in Economics, together with John Nash and Reihard Selten, mainly because of this formulation.

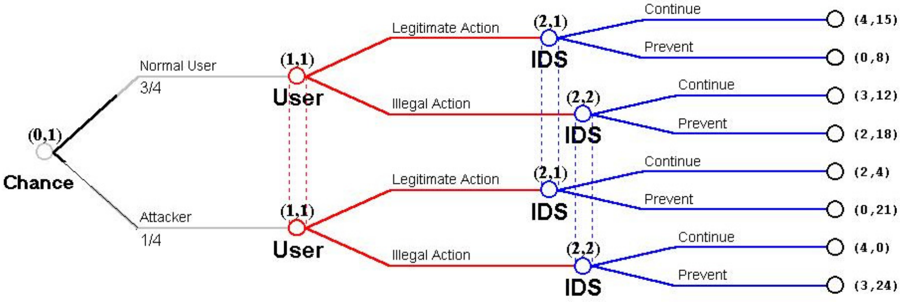

Fig. 1. Intrusion Detection as a signaling game

They model strategic problems in which the players have no complete information about each other's preferences. They also have the potential to model irrationality, as was shown in the famous "gang of four" paper in 1982 (Kreps, Milgrom, Roberts, and Wilson).

3 Constructing the Normal Form of the ID Signaling Game

In the ID game there are two players, the *IDS* which protects the Target System and a *User* who uses the Target System. In reality, there are a number of users who act on the Target System, but this is a more complicated setting. We assume that player *User* has a binary choice between two actions: he can either act Legitimately (L) or he can act Illegally (I). These actions are the signals sent to the *IDS* from the *User*. The *IDS* has also two actions, to Prevent (P) the *User* using the Target System or to permit him to Continue (C), because it decided that the signals come from an *Attacker* or from a *Normal User* respectively.

The possible actions described above lead to the sets of strategies that correspond to each player. Two capital letters are assigned to each strategy, the first corresponds to an action when the *User* is a *Normal User*, and the second corresponds to an action when the *User* is an *Attacker*. So, the *User* has four strategies. First, he can act Legitimately regardless he is a *Normal User* or he is an *Attacker* (LL). He can act Legitimately if he is a *Normal User* and Illegally if he is an *Attacker* (LI). He can act Illegally if he is a *Normal User* (accidentally) and Legitimately if he is an *Attacker* (bluffing) (IL). Ultimately, he can act Illegally no matter what he is (II).

The *IDS* has four strategies too. It can Prevent the *User* whatever he is (including false positives) (PP). It can Prevent the *User* if he is an *Attacker* and allow him to Continue if he is a *Normal User* (CP). It can allow the *User* to

Continue if he is an *Attacker* (false negatives) and Prevent the *User* if he is a *Normal User* (false positives) (*PC*). Finally, it can allow the *User* to Continue regardless he is a *Normal User* or an *Attacker* (including false negative) (*CC*). It is remarkable that all these strategies encompass false alarms except the second one which is the optimal case, to allow a *Normal User* to Continue and to Prevent an *Attacker*. Besides, strategy *PC* seems irrational, but in fact, in this case the *IDS* does not trust the signals it gets from the *User*.

The payoffs assigned to each strategy are summarized in Table 4 below, where both cases of a *Normal User* or an *Attacker* are included. Rows correspond to *User*'s strategies and columns to the *IDS*'s strategies.

Table 4. IDS's Utility Function in a Signaling Game

	PP	CP	PC	CC
LL	(0,8)/(0,21)	(4,15)/(0,21)	(0,8)/(2,4)	(4,15)/(2,4)
LI	(0,8)/(3,24)	(4,15)/(3,24)	(0,8)/(4,0)	(4,15)/(4,0)
IL	(2,18)/(0,21)	(3,12)/(0,21)	(2,18)/(2,4)	(3,12)/(2,4)
II	(2,18)/(3,24)	(3,12)/(3,24)	(2,18)/(4,0)	(3,12)/(4,0)

Each cell includes a couple of payoffs pairs. The first pair corresponds to the case of a *Normal User* and the second pair corresponds to the case of an *Attacker*. The first number in each pair is *User*'s payoff and the second is *IDS*'s payoff. In Figure 2 we zoom at the matrix for details.

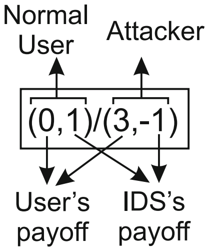

Fig. 2. Details of the notation used in a cell of the payoffs matrix

Because the *IDS* knows with probability $\frac{1}{4}$ that the *User* is an *Attacker* and with probability $\frac{3}{4}$ that he is a *Normal User*, the expected return to the players should be calculated by adding the first half of the matrix multiplied by $\frac{3}{4}$ and the second half of the matrix multiplied by $\frac{1}{4}$. The calculations are given in the sequence:

$$\frac{3}{4}\cdot\begin{pmatrix} (0,8) & (4,15) & (0,8) & (4,15) \\ (0,8) & (4,15) & (0,8) & (4,15) \\ (2,18) & (3,12) & (2,18) & (3,12) \\ (2,18) & (3,12) & (2,18) & (3,12) \end{pmatrix} +$$

$$\frac{1}{4}\cdot\begin{pmatrix} (0,21) & (0,21) & (2,4) & (2,4) \\ (3,24) & (3,24) & (4,0) & (4,0) \\ (0,21) & (0,21) & (2,4) & (2,4) \\ (3,24) & (3,24) & (4,0) & (4,0) \end{pmatrix} =$$

$$\begin{pmatrix} (0,6) & (3,11\frac{1}{4}) & (0,6) & (3,11\frac{1}{4}) \\ (0,6) & (3,11\frac{1}{4}) & (0,6) & (3,11\frac{1}{4}) \\ (1\frac{1}{2},13\frac{1}{2}) & (2\frac{1}{4},9) & (1\frac{1}{2},13\frac{1}{2}) & (2\frac{1}{4},9) \\ (1\frac{1}{2},13\frac{1}{2}) & (2\frac{1}{4},9) & (1\frac{1}{2},13\frac{1}{2}) & (2\frac{1}{4},9) \end{pmatrix} +$$

$$\begin{pmatrix} (0,5\frac{1}{4}) & (0,5\frac{1}{4}) & (\frac{1}{2},1) & (\frac{1}{2},1) \\ (\frac{3}{4},6) & (\frac{3}{4},6) & (1,0) & (1,0) \\ (0,5\frac{1}{4}) & (0,5\frac{1}{4}) & (\frac{1}{2},1) & (\frac{1}{2},1) \\ (\frac{3}{4},6) & (\frac{3}{4},6) & (1,0) & (1,0) \end{pmatrix} =$$

$$\begin{pmatrix} (0,11\frac{1}{4}) & (3,16\frac{1}{2}) & (\frac{1}{2},7) & (3\frac{1}{2},12\frac{1}{4}) \\ (\frac{3}{4},12) & (3\frac{3}{4},17\frac{1}{4}) & (1,6) & (4,11\frac{1}{4}) \\ (1\frac{1}{2},18\frac{3}{4}) & (2\frac{1}{4},14\frac{1}{4}) & (2,14\frac{1}{2}) & (2\frac{3}{4},10) \\ (2\frac{1}{4},19\frac{1}{2}) & (3,15) & (2\frac{1}{2},13\frac{1}{2}) & (3\frac{1}{4},9) \end{pmatrix}$$

To avoid having payoffs in a fraction format, we multiply the above matrix by 4, and we get the following final payoff matrix:

Table 5. Payoff Matrix for the ID Signaling Game

	PP	CP	PC	CC
LL	0,45	12,66	2,28	14,49
LI	3,48	15,69	4,24	16,45
IL	6,75	9,57	8,58	11,40
II	9,78	12,60	10,54	13,36

4 Removing Dominated Strategies

We first solve the ID signaling game by applying the domination criterion, which says that a rational player should not use a dominated strategy. Binmore [1] expresses the domination criterion by assuming two strategies s_1 and s_2 of a player *I* and three strategies t_1, t_2, and t_3 of a player *II.* Then we decide that for player *I*, strategy s_2 strongly dominates strategy s_1 when

$$\pi_1(s_2, t) > \pi_1(s_1, t) \tag{5}$$

holds for all three values of player *II*'s strategy t. Moreover, if the relation between two strategies is $\geq$, then the one strategy weakly dominates the other. In our game we express in algebraic terms the above criterion to check if it holds. First, we consider that *IL* is dominated by *II* because:

```
[6,9,8,11] < [9,12,10,13]
```

Using this domination argument, we remove strategy *IL* from the payoff matrix and the matrix changes to the following:

	PP	CP	PC	CC
LL	0,45	12,66	2,28	14,49
LI	3,48	15,69	4,24	16,45
II	9,78	12,60	10,54	13,36

Second, *PC* is dominated by *PP* because:

```
[45,48,78] > [28,24,54]
```

and thus we reduce the payoff matrix again by removing strategy *PC*. The payoff matrix now has the following form:

	PP	CP	CC
LL	0,45	12,66	14,49
LI	3,48	15,69	16,45
II	9,78	12,60	13,36

Third, *LL* is dominated by *LI* which can be expressed in algebraic terms as

```
[3,15,16] > [0,12,14]
```

So, the strategy *LL* is also out of the matrix and the payoff matrix turns into the following:

	PP	CP	CC
LI	3,48	15,69	16,45
II	9,78	12,60	13,36

Finally, strategy *CC* is dominated by strategy *CP* as

```
[69,60] > [45,36]
```

Consequently, the above deletions lead to a smaller $2x2$ matrix as shown in Table 6.

Table 6. Reduced Payoff Matrix for the ID Signaling Game

	PP	CP
LI	3,48	15,69
II	9,78	12,60

Studying the resulting matrix, it makes sense that a *Normal User* can either act Legitimately or Illegally, while an *Attacker* acts only Illegally. In addition, we should check if there is a mixed strategy equilibrium. Consider that the probability for player *User* of playing strategy *LI* is p, and the probability of playing strategy *II* is $1-p$. Then, because the *IDS*'s payoffs in strategies *PP* and *CP* are 48 points in the *LI* strategy, 78 points in the *II* strategy, and 69 points in the *LI* strategy and 60 points in the *II* strategy respectively, we get the following equation:

$$48p + 78(1-p) = 69p + 60(1-p) \tag{6}$$

Solving Equation 6 to determine p, we get $p = \frac{18}{39}$ which is very close to 0.5 ($\simeq 0.461538$). The inference is that there is no Nash equilibrium for which player *User* will decide to play the *LI* strategy. It sounds reasonable that an *Attacker* will think about acting Illegally all the time and that a *Normal User* makes mistakes[2].

Similarly, consider that p' is the probability that player *IDS* will choose strategy *PP*, and $1-p'$ is the probability player *IDS* will choose strategy *CP*. Then the following equation must hold:

$$3p' + 15(1-p') = 9p' + 12(1-p') \tag{7}$$

Solving Equation 7 to determine p', we get $p' = \frac{1}{3}$. Therefore, player *IDS* will Prevent the *User* to Continue with probability $\frac{1}{3}$ regardless he is a *Normal User* or an *Attacker*. Furthermore, the *IDS* will let a *Normal User* to Continue and Prevent an *Attacker* with probability $\frac{2}{3}$ which is the most rational strategy.

It seems that player *User* is indifferent between strategy *LI* and strategy *II*. Therefore, the results lead our reasoning to different approaches, as those described in the following section.

5 Computing Equilibria in the ID Signaling Game

In a signaling game with two players, the one player knows something the other doesn't, that is, the one player holds information the other doesn't, but he sends

[2] A *Normal User* who is acting legitimately for a while, but at the next point of time accidentally acts illegally, could be modeled with Selten's "Trembling-Hand Equilibrium".

signals to give hints of this private information to the second player. When the other player picks up the signal, then he decides upon this what action to take as a response. The corresponding (assigned) payoffs show the winner and the loser of the game. A signaling game usually admits more than one Nash equilibria.

The Intrusion Detection game described previously is an example of a signaling game, because player *User* knows if he is a *Normal User* or an *Attacker*, whereas the Intrusion Detection System doesn't. In addition, the *User* sends signals to the *IDS* by using the Target System, the *IDS* collects the events generated by this activity and decides whether to prevent or to allow the *User* to continue using the TS, by judging if this activity belongs to a *Normal User* or to an *Attacker*. In the Intrusion Detection game, *Chance* should start the game by deciding if player *User* is a *Normal User* or an *Attacker*. Then, player *IDS* is at the opposite side of the *User* and it might either Prevent the *User* or it might allow the *User* to Continue using the TS. The *IDS* would Prevent the *User* if it were aware that the *User* would damage the system, i.e. he is an *Attacker*, and it would allow the *User* to Continue if it were aware that no damage would be caused, i.e. he is a *Normal User*. Unfortunately, only the *User* knows for sure that he is a *Normal User* or an *Attacker*. In other words, only the *User* knows his type.

However, by using the TS, the *User* is sending a signal of Legitimate activity when he is a *Normal User*, and a signal of Illegal activity when he is an *Attacker*. To determine the type of signal that corresponds to each activity, i.e. to decode a signal, we assume that the event reception module hosted by the *IDS*, collects the lowest level functions of the operating system (e.g. system calls) and examines their return values. If the return value of a function indicates that the User has attempted a system violation, or a security relevant event has successfully taken place, then an illegal activity is assigned with this. Otherwise, a legitimate action has taken place.

On average, a *Normal User* will act Legitimately and an *Attacker* Illegally. Nevertheless, a *Normal User* might act accidentally Illegally, because for instance he is a novice and as such he makes mistakes (Selten's trembling hand perfect equilibrium). Likewise, an *Attacker* might act Legitimately as an attempt to bluff so he can avoid detection, or because he is an insider, so he is authorized for a number of activities, but he takes advantage of them to cause damage. In conclusion, the *User* who sends a signal might confuse the *IDS* on purpose or unintentionally.

In the next subsections, we follow Binmore's reasoning for the quiche game [1] and Gintis's concepts [3], to solve the ID game by locating any Nash equilibria, first in pure strategies and afterwards in behavioral strategies.

5.1 Locating Nash Equilibria in Pure Strategies

First, we examine the ID signaling game for Nash equilibria in pure strategies. Assume that player *IDS* chooses strategy *PP*. Then, the best response for player *User* is strategy *IL*, because it is reasonable to play with Legitimate actions if he is a *Normal User* and to act Illegally if he is an *Attacker*, not bluffing since he will be caught anyway. Besides, the payoff matrix shows that he is loosing less by playing *IL* than in any other choice. Considering the other way, if player *User* chooses *IL*, then the best response for player *IDS* is again *PP*. Therefore, the pair of strategies *IL* and *PP* is a Nash equilibrium.

Next, assuming that player *IDS* plays strategy *PC*, then player *User* plays *II* as his best response. But reversing the argument, shows that, if player *User* plays *II*, then player *IDS* plays *PP* and not *PC*, because his payoff is maximized with *PP* (6 points instead of 1). So, there are no Nash equilibria in which player *IDS* chooses strategy *PC*.

Similarly, if player *User* uses strategy *LL*, then player *IDS*'s best reply is strategy *CP*, whereas, if player *IDS* plays strategy *CP*, then player *User* will choose either strategy *LL* or strategy *IL* as best reply, because their payoffs are equal (5 points). Namely, it is undetermined what player *User* will do; he will act Illegally or Legitimately in the case he is a *Normal User*? Still, player *IDS* should counteract if player *User* acts Illegally and Prevents him from damaging the TS. There is a point here that requires further consideration.

In certain environments, we care not only about information related to the knowledge of the players, but also about information related to their beliefs. In our case, we examine player *IDS*'s beliefs after receiving a signal from player *User*, i.e. collects an event from the TS generated by the *User*. It is coherent for player *IDS* to allow the *User* to Continue using the TS, if it gets a signal of *Normal User*, and to Prevent him if it gets a signal of *Attacker* from him. So, the fact that player *IDS* chooses *CP* adds no more information. If the initial probability that player *User* is a *Normal User* is p, and that he is an *Attacker* is $1 - p$, this remains unchanged. However, the payoffs lead player *User* to act Legitimately when the probability of being a *Normal User* is higher. For that reason, it is XL (i.e. *IL* or *LL*) the best reply to strategy *CP*. As a result, there are no Nash equilibria in which player *IDS* chooses strategy *CP*.

Following the same reasoning, consider that player *IDS* uses strategy *CC*. Consequently, player *User* will use one of the strategies *LI* or *II*, that is, it is undetermined if a *Normal User* will act Illegally or Legitimately, whereas an *Attacker* will definitely act Illegally because he will evade detection. Reversing the case, if player *User* chooses *LI*, then the best response for player *IDS* is not strategy *CC* but strategy *CP* (3 points instead of -2). The conclusion is that there are no Nash equilibria when player *IDS* chooses strategy *CC*.

To end with the pure strategies, there is only one Nash equilibrium in pure strategy *IL* for player *User* and strategy *PP* for player *IDS*. Verifying this finding, the *IDS* Prevents a *Normal User* when he is acting Illegally either by purpose or unintentionally, but it prefers also to Prevent an *Attacker* to continue using the TS when he is acting Legitimately, because this legal activity might form the first steps of a complete attack scenario.

Although a Nash equilibrium has been located in pure strategies, it is necessary to look for other Nash equilibria in mixed strategies. Such a task is quite difficult and complicated, but Nash has proved that *every finite game has at* *least one Nash equilibria if mixed strategies are allowed* [1]. In any case, we will achieve valuable results upon completion.

In order to facilitate this work, one can replace mixed strategies by behavioral strategies. In the next section, there is an explanation why this can be done, and a description of the behavioral strategies search for Nash equilibria.

5.2 Locating Nash Equilibria in Behavioral Strategies

Perfect recall games are those where no player ever forgets any piece of information that was once in his knowledge. Thus, the ID game is a perfect recall game. In addition, Kuhn has proved the following theorem for perfect recall games [1]:

> *Kuhn's theorem*
> Whatever mixed or behavioral strategy s that player i may choose in a game of perfect recall, he or she has a strategy t of the other type with the property that, however the opponents play, the resulting lottery over the outcomes of the game is the same for both s and t.

With this theorem, Kuhn has proved that in perfect recall games, mixed strategies and behavioral strategies are identical. Therefore, instead of searching for Nash equilibria in mixed strategies, we will examine the behavioral strategies of the ID game.

A behavioral strategy is a decentralized mixed strategy, that is, like a pure strategy, it is clear for a player what action to take at each information set, but, a probability is assigned to each action [1]. Based on this probability, a player decides how to proceed the game.

In our game, for player *User*, a behavioral strategy must assign a probability p with which the *User* will act Legitimately at the information set *Normal User*, and a probability q with which player *User* will act Legitimately at the information set *Attacker*. Correspondingly, the probability with which player *User* will act Illegally at the information set *Normal User* is $1 - p$, and the probability with which player *User* will act Illegally at the information set *Attacker* is $1 - q$.

Similarly, considering player *IDS*'s behavioral strategies, a probability r must be assigned to the action Prevent at the information set *Illegal Activity*, and a probability s to the action Prevent at the information set *Legitimate Activity*. The probabilities $1 - r$ and $1 - s$ must be assigned to the action Continue, at the information sets *Illegal Activity* and *Legitimate Activity*, respectively.

The established probabilities affect the initial probabilities with which the game starts. In particular, we mentioned before that at the beginning of the game, player *IDS* knows with probability $\frac{1}{4}$ that player *User* is an *Attacker*, and with probability $\frac{3}{4}$ that he is a *Normal User*. This is said to be the player's prior probabilities for the events that the *User* is an *Attacker* or a *Normal User*, respectively. Now, another piece of information is added to the *IDS*'s knowledge. It is the behavioral strategy (p, q), which represents the case of acting Legitimately, whatever player *User* is (*Normal User* or *Attacker*). If player *IDS* knows the probabilities p and q, because someone wrote them in a game theory book as Binmore says, it should update its beliefs about what player *User* is. These new probabilities are called posterior probabilities, and the process we follow from prior to posterior probabilities is called Bayesian updating.

Assuming that player *User* chooses to play strategy (p, q), the probability that the upper branch will be followed and node (a) will be reached is $\frac{3}{4} * p$ whereas the probability to reach the corresponding node (b) is $\frac{1}{4} * q$. Thus, at the information set *Legitimate Activity*, the posterior probability for player *IDS* when the *User* is a *Normal User*, is

$$\text{prob(User is Normal|User acts Legitimately)} = \frac{prob(a)}{prob(a)+prob(b)} = \frac{\frac{3}{4}*p}{\frac{3}{4}*p+\frac{1}{4}*q}$$

and when the *User* is an *Attacker*, is

$$\text{prob(User is Attacker|User acts Legitimately)} = \frac{prob(b)}{prob(a)+prob(b)} = \frac{\frac{1}{4}*q}{\frac{3}{4}*p+\frac{1}{4}*q}$$

Analyzing player *IDS*'s behavior first at the information set *Legitimate Activity*, we take into account that player *IDS* prefers to Prevent player *User*, when the latter is an *Attacker*. Since the probability at the information set *Legitimate Activity* is $\frac{3}{4}*p$ for a *Normal User* and $\frac{1}{4}*q$ for an *Attacker*, the *IDS* will Prevent the *User* at this node of the game, if the following inequality holds:

$$\frac{1}{4}q > \frac{3}{4}p \Rightarrow q > 3p \tag{8}$$

On the other hand, player *IDS* will allow the *User* to Continue at the information set *Legitimate Activity*, if the reverse inequality holds, that is,

$$q < 3p \tag{9}$$

Finally, player *IDS* has no interest in choosing either to Prevent or to allow the *User* to Continue, if the probabilities are equal, that is,

$$q = 3p \tag{10}$$

Regarding the *IDS*'s behavior at the information set *Illegal Activity*, the player *IDS* will choose to Prevent the *User* from using the TS if the following inequality holds:

$$\frac{1}{4}(1-q) > \frac{3}{4}(1-p) \tag{11}$$

Simplifying the inequality (11) we get

$$q > 3p - 2 \tag{12}$$

Similarly, player *IDS* will allow the *User* to Continue at the information set *Illegal Activity*, if the reverse inequality holds, that is,

$$q < 3p - 2 \tag{13}$$

Finally, player *IDS* has no interest in choosing either to Prevent or to allow the *User* to Continue when his signals indicate *Illegal Activity*, if the probabilities are equal, that is,

$$q = 3p - 2 \tag{14}$$

When we examined the existence of Nash equilibria in pure strategies, we faced the case of undetermined choices. In particular, we found that if player *IDS* plays strategy *CP*, then player *User* will choose either strategy *LL* or strategy *IL* as best reply, because their payoffs are equal (5 points). That is to say, we do not know what player *User* will do at this node of the game. As a *Normal User*, he will either act *Illegally* or *Legitimately*, and he is apathetic in choosing whichever strategy. This was the reason we switched to behavioral strategies, in order to determine all Nash equilibria. The equations (10) and (14) reveal such cases.

Assuming that the hypothesis (14) is true, then it is also true that

$$q < 3p \tag{15}$$

So, the conclusion that derives from (15) is that, player *IDS* will allow the *User* to Continue at the information set *Legitimate Activity*. Consequently, there is no point for player *User* to act Legitimately when he is an *Attacker*, so he will better decide to act Illegally. Besides, this might be closer to his aims and temperament. Therefore, it should be $1 - q = 1$, which results in $q = 0$. But then probability p can be calculated by (14), that is $p = \frac{2}{3}$. As a result, there is a Nash equilibrium with $q = 3p - 2$, when $q = 0$ and $p = \frac{2}{3}$.

To make it meaningful, player *User* will decide to act Legitimately with probability $\frac{2}{3}$ if he is a *Normal User*, while he will definitely decide to act Illegally with probability 1 if he is an *Attacker*. Moreover, player *User* will play Illegally with probability $\frac{1}{3}$ if he is a *Normal User*.

Furthermore, the following equation must hold when examining player *IDS*'s behavior at the information set *Attacker*:

$$(-2)r + 4(1 - r) = 3 - 2 \tag{16}$$

Solving Equation 16 we get $r = \frac{1}{2}$. Consequently, the next equation must also hold:

$$(-1)s + 3(1 - s) = 2 - 1 \tag{17}$$

Solving also Equation 17 we get $s = \frac{1}{2}$.

Likewise, at the information set *Normal User*, the following equation must hold:

$$0r + 2(1 - r) = 1 - 1 \tag{18}$$

Solving Equation 18 we get $r = 1$. Consequently, the next equation must also hold:

$$(-1)s + 2(1 - s) = -1 + 0 \tag{19}$$

Solving also Equation 19 we get $s = 1$.

Decoding these findings, we realize that the *IDS*'s behavior is indifferent between Preventing an *Attacker* from acting either Illegally or Legitimately ($r = s = \frac{1}{2}$). In addition, it sounds strange that the *IDS* will Prevent a *Normal User* to continue using the TS for sure, when acting either Illegally or Legitimately. All these happen when assuming that initially player *User* is a *Normal User* with probability $\frac{3}{4}$ or an *Attacker* with probability $\frac{1}{4}$. If the initial probabilities change, then the above results will be affected significantly. Specifically the probability with which a *Normal User* acts Illegally ($\frac{1}{3}$) will be decreased, if we decrease the initial probability $\frac{1}{4}$ with which a *User* is an *Attacker*. This is really high to be true.

Next, if the hypothesis (10) is true, then the inference is that player *IDS* has no interest in deciding either to Prevent or to allow the *User* to Continue, if the probabilities are equal. But from Inequality 15 we know that there is a Nash equilibrium when $q < 3p$. Therefore, there is no way to have another Nash equilibrium when $q = 3p$.

6 Related Work

Patcha and Park have modeled the interactions between the nodes of an ad hoc network as an incomplete information game in [6] and [7]. They formulate a signaling game between an attacker and a node of a MANET, where an IDS is present to defend attacks. They use a more general perspective in their approach, they assume that a node might be either a regular node or a malicious node/attacker, but they do not use the utility function nor payoffs calculated by players' preferences. Although their approach is interesting, it is not possible to get insights from the structure of players' behavior in such a generalized framework.

Among the most recent related works is described in [4]. Lin et.al. have used signaling games to model specific attack-defense scenarios on confidentiality. Their work assumes that an IDS is a defender and examines which strategies are maximizing its payoffs when detection has been completed.

In [8] there is another signaling game for wireless sensor networks, in which some nodes are malicious. An intrusion detection mechanism is proposed to assist the selection of optimal strategies to defend these malicious nodes.

7 Conclusion

The proposed work provides a formulation of the ID game as a signaling game, by examining players' preferences and calculating the corresponding payoffs. The construction of such a game requires the same elements to be specified, but the game has a different method to be examined and solved. Our results show that the problem of multiple NE and which one to choose appears again revealing the uncertainty of how the game will be played in the future. Therefore, there is a need for Nash equilibrium refinements, in order to choose one that might be selected. But also, the ID signaling game shows the need for defining new signals, which will support its formulation and will give us a better understanding in the interactions that take place between attackers and IDSs. Rather than trying again to solve the problem of signal identification, we have chosen an approach for behavioral detection, given that an IDS identifies with a certain accuracy a user's actions. This is the inherent uncertainty of an ID signaling game. Future research directions include many users who act on the Target System, but this is a more complicated setting.

References

1. Binmore, K.: Playing for Real - A Text on Game Theory. Oxford University Press (2007)
2. McKelvey, R.D., McLennan, A.M., Turocy, T.L.: Gambit: Software Tools for Game Theory, Version 0.2007.01.30 (January 2007), http://www.gambit-project.org (accessed December 20, 2012)
3. Gintis, H.: Game Theory Evolving - A Problem-Centered Introduction to Modeling Strategic Interaction. Princeton University Press (2000)
4. Lin, J., Liu, P., Jing, J.: Using Signaling Games to Model the Multi-step Attack-Defense Scenarios on Confidentiality. In: Grossklags, J., Walrand, J. (eds.) GameSec 2012. LNCS, vol. 7638, pp. 118–137. Springer, Heidelberg (2012)
5. Osborn, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press (1994)
6. Patcha, A., Park, J.-M.: A Game Theoretic Approach to Modeling Intrusion Detection in Mobile Ad Hoc Networks. In: Proc. of the 2004 IEEE Workshop on Information Assurance and Security (June 2004)
7. Patcha, A., Park, J.-M.: A Game Theoretic Formulation for Intrusion Detection in Mobile Ad Hoc Networks. International Journal of Network Security 2(2), 131–137 (2006)
8. Shen, S., Li, Y., Xu, H., Cao, Q.: Signaling game based strategy of intrusion detection in wireless sensor networks. Computers and Mathematics with Applications (62), 2404–2416 (2011)
9. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press (1953)

Author Index