Andreas U. Schmidt
Shiguo Lian (Eds.)

LNICST

17

Security and Privacy in Mobile Information and Communication Systems

First International ICST Conference, MobiSec 2009
Turin, Italy, June 2009
Revised Selected Papers

ICST

Springer

Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 17

Andreas U. Schmidt   Shiguo Lian (Eds.)

Security and Privacy in Mobile Information and Communication Systems

First International ICST Conference, MobiSec 2009
Turin, Italy, June 3-5, 2009
Revised Selected Papers

Springer

Volume Editors

Andreas U. Schmidt
novalyst IT AG
Robert-Bosch Str. 38
61184 Karben, Germany
E-mail: andreas.schmidt@novalyst.de

Shiguo Lian
France Telecom R&D
10F, South Twr., Raycom Infotech, Park C, 2
Science Institute South Rd.
Haidian District, Beijing 100080, China
E-mail: shiguo.lian@orange-ftgroup.com

Library of Congress Control Number: 2009935010

CR Subject Classification (1998): C.2, D.4.6, K.6.5, K.4.4, C.1.3, K.4.1

ISSN 1867-8211
ISBN-10 3-642-04433-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-04433-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12761173 06/3180 5 4 3 2 1 0

Preface

MobiSec 2009 was the first ICST conference on security and privacy in mobile information and communication systems. Within the vast area of mobile technology research and application, the intention behind the creation of MobiSec was to make a small, but unique, contribution. Our aim in conceiving this conference was to build a bridge between top-level research and large-scale application of novel kinds of information security for mobile devices and communication. It has been a privilege to serve this event as a General Chair.

In this, first, year, MobiSec already attracted some excellent scientific papers, application studies, and contributions from industrial research from all over the world. Apart from that, the conference emphasis was on European leadership, by featuring a keynote from the European Commission, and contributions from the European Commssion-funded research projects SWIFT and OpenTC, which exhibited the strength of European industry and academia in the field. MobiSec's focus on knowledge transfer was supported by a tutorial on the Host Identification Protocol—a very topical contribution, which emphasizes the need to build in security in the autonomous, unsupervised parts of networking, in a cross-layer approach.

The papers at MobiSec 2009 dealt with a broad variety of subjects ranging from issues of trust in and security of mobile devices and embedded hardware security, over efficient cryptography for resource-restricted platforms, to advanced applications such as wireless sensor networks, user authentication, and privacy in an environment of autonomously communicating objects. With hindsight a leitmotif emerged from these contributions, which corroborated the idea behind MobiSec. A set of powerful tools have been created in various branches of the security discipline, which await combined application to build trust and security into mobile (that is, all future) networks, autonomous and personal devices, and pervasive applications.

Many people contributed to the success of MobiSec. It was a privilege to work with these dedicated persons, and I would like to thank them all for their efforts. The Organizing Committee as a whole created a frictionless, collaborative work atmosphere, which made my task an easy one. Antonio Lioy, who also did a lot of the local organization together with Daniele Mazzocchi, and Neeli Prasad did a great job assembling the Technical Program Committee and running the paper submission and review process. A high-quality conference cannot be created without the help of the distinguished members of the Program Committee—the soul of each scientific event. Shiguo Lian meticulously oversaw the preparation of the conference proceedings. The keynote speech by Bart van Caenegem of the EU Commission is very much appreciated, as well as the presentations by exponents of the EU research projects. Andrei Gurtov contributed a tutorial on an important and current topic. Special credit is due to Gergely Nagy, our conference coordinator from ICST, and his colleagues there, for their excellent support during the preparation of the event. It is hard to find an administrator under this workload who shows such responsiveness, competence, and pragmatic attitude. Beatrix Ransburg and Eszter Hajdu oversaw the compilation of the CD ROM proceedings and this proceedings volume with calm and professionalism. Finally, I thank Imrich Chlamtac and the members of the Steering Committee for their support and guidance during these months and for entrusting me with the task of chairing MobiSec 2009.

Organizing a conference which is the first of its kind is always an adventure, if not venturous in the turbulent times we live in. We are delighted to have been part of it. Concluding, I am confident that all participants found MobiSec a stimulating and thought-provoking event, and enjoyed their time in Turin. We are looking forward to next year's MobiSec.

July 2009 Andreas U. Schmidt

Organization

Steering Committee Chair

Imrich Chlamtac	President, Create-Net Research Consortium, Trento, Italy

Steering Committee Members

Andreas U. Schmidt	Create-Net Research Consortium, Trento, Italy
Tibor Kovacs	Director of Business and Technology Affairs, ICST

General Chair

Andreas U. Schmidt	Create-Net Research Consortium, Trento, Italy

Technical Program Co-chairs

Neeli R. Prasad	Aalborg University, Denmark
Antonio Lioy	Politecnico di Torino, Italy

Local Arrangements Co-chairs

Daniele Mazzochi	Istituto Superiore Mario Boella (ISMB), Turin, Italy
Flaminia Luccio	University Ca'Foscari, Venice, Italy

Workshops Co-chairs

Rüdiger Grimm	University of Koblenz-Landau, Germany
Shiguo Lian	France Telecom R&D, Beijing, China

Panels Chair

Seung Woo Seo	Seoul National University, Korea

Publications Chair

Shiguo Lian	France Telecom R&D, Beijing, China

Web Chair

Giovanni Russello	Create-Net Research Consortium, Trento, Italy

Conference Coordinator

Gergely Nagy	ICST

Technical Program Committee

Selim Aissi	Intel, USA
Mahbubul Alam	Cisco, USA
Francesco Bergadano	Università degli Studi di Torino, Italy
Inhyok Cha	InterDigital Communications, USA
Rocky K. C. Chang	Hong Kong Polytechnic University, China
Hsiao-Hwa Chen	National Sun Yat-Sen University, Taiwan
Shin-Ming Chen	National Taiwan University, Taiwan
Tassos Dimitriou	AIT, Greece
Loic Duflot	SGDN/DCSSI, France
Ashutosh Dutta	Telcordia, USA
Stefanos Gritzalis	University of the Aegean, Greece
Markus Gueller	Infineon, Germany
Rajesh Gupta	University of California San Diego, USA
Marco Hauri	ASCOM, Switzerland
Mario Hoffmann	Fraunhofer SIT, Darmstadt, Germany
Jiankun Hu	RMIT University, Australia
Kazukuni Kobara	AIST, Japan
Geir Myrdahl Koien	Telenor, Norway
Shiguo Lian	France Telecom R&D, Beijing, China
Flaminia Luccio	Università Ca' Foscari, Venice, Italy
Fabio Martinelli	IIT and CNR, Pisa, Italy
Hassnaa Moustafa	France Telecom R&D (Orange Labs), France
Valtteri Niemi	Nokia, Finland
Vladimir Oleshchuk	UIA, Norway
Max Ott	NICTA, Australia
Jong Hyuk Park	Kyungnam University, Korea
Christos Politis	University of Kingston, UK
Anand Prasad	NEC Research Laboratories, Japan
Yi Qian	NIST, USA
Reijo Savola	VTT, Finland
Georg Schaathun	Surrey University, UK
Jean-Pierre Seifert	Samsung, USA
Yogendra Shah	InterDigital Communications
Chris Swan	Credit Suisse IT R&D, UK
Krzysztof Szczypiorski	Warsaw University of Technology, Poland
Allan Tomlinson	Royal Holloway University of London, UK
Janne Uusilehto	Nokia, Finland
Anna Vaccarelli	IIT and CNR, Pisa, Italy
Xin Wang	ContentGuard Inc., USA
Zheng Yan	Nokia Research Center, Finland

Table of Contents

On Trust Evaluation in Mobile Ad Hoc Networks

Dang Quan Nguyen[1], Louise Lamont[1], and Peter C. Mason[2]

[1] Communications Research Centre, Canada
{Dang.Nguyen,Louise.Lamont}@crc.gc.ca
[2] Defence Research & Development, Canada
Peter.Mason@drdc-rddc.gc.ca

Abstract. *Trust* has been considered as a social relationship between two individuals in human society. But, as computer science and networking have succeeded in using computers to automate many tasks, the concept of *trust* can be generalized to cover the reliability and relationships of non-human interaction, such as, for example, information gathering and data routing. This paper investigates the evaluation of trust in the context of ad hoc networks. Nodes evaluate each other's behaviour based on observables. A node then decides whether to trust another node to have certain innate abilities. We show how accurate such an evaluation could be. We also provide the minimum number of observations required to obtain an accurate evaluation, a result that indicates that observation-based trust in ad hoc networks will remain a challenging problem. The impact of making networking decisions using trust evaluation on the network connectivity is also examined. In this manner, quantitative decisions can be made concerning trust-based routing with the knowledge of the potential impact on connectivity.

Keywords: Ad hoc networks, security, trust evaluation, connectivity.

1 Introduction

A Mobile ad hoc network (MANET) consists of auto-configuring nodes that communicate with each other using wireless equipment. Such networks are infrastructureless, self-deploying and do not require a centralized entity. These advantages make MANETs suitable for critical uses: tactical military networks, disaster recovery, etc. Messages between two out-of-range nodes are routed in a multi-hop way, through intermediate nodes selected by MANET routing protocols (e.g. OLSR [1]).

These characteristics have a deep impact on security issues as they pose new challenges to the design of security solutions. One of these issues is concerned with trust management. The ad hoc environment is distributed and changing, meaning nodes can join and leave the network at any time. Therefore, traditional identification schemes based on a centralized authentication server are generally unsuitable for ad hoc networks. Ad hoc networks require new trust management designs to support a distributed environment and to be more robust against topology changes. One of the main components of trust management is *trust* *evaluation*, or how to estimate the degree of trust between two nodes in an ad hoc network.

In this paper, we focus on the evaluation of trust. We define trust in the context of ad hoc networks and show how observations can be used to build an accurate estimate of trust. We also investigate the effect of decisions, based on trust evaluation, on one of the important properties of ad hoc networks which is connectivity. Indeed, trust decisions based on strict selection policy may result in few nodes selected, thus the network topology made of trusted nodes may be disconnected.

The remainder of this paper is organized as follows. In Section 2, we describe the existing work on trust computation. We start our study by giving a new definition of trust in ad hoc networks in Section 3 and discuss some aspects of this definition. Trust evaluation and estimation accuracy are explained in Section 4. We investigate the effect of trust evaluation on the network connectivity in Section 5. We conclude this paper in Section 6.

2 Related Work

There has been a significant amount of work done on trust. One of the earliest results on the topic within computer science [2] shows that trust can be formalized as a computational concept. Since then, research on trust has evolved in two main directions: *trust evaluation* and *trust sharing.*

Trust evaluation is concerned with the problem of estimating the trustworthiness of an entity (called a *node*) within a system, usually viewed as a network of interacting nodes. In [3], the authors propose a model for trust computation. This model defines trust as a subjective expectation a node has about another node based on the history of their encounters. This definition is probably closest to the one we will present. However, the work done in [3] is limited in the sense that it only takes into account a node's binary actions (cooperate or defect)– that is, the trust is discrete.

In [4], the authors use entropy to measure the uncertainty in trust relationship. This entropy is obtained from the probability that a node will perform some action. Such a probability is useful because it can be used as factor in predicting the behaviour of a node; that is, it can be used to estimate its trustworthiness. The authors do not, however, specify how this probability of node's compliance is arrived at in the first place.

On the other hand, trust sharing is concerned with the problem of sharing the estimation of a node's trustworthiness with other (usually distant) nodes and, conversely, of synthesizing all the received estimations. In [5], the authors propose an algorithm allowing indirect neighbours to estimate the trustworthiness of each other based on the trustworthiness of direct neighbours, as long as there exists a path between them. This algorithm of trust sharing treats it as a single real value between 0 and 1. It also assumes that trust propagation is multiplicative. Thus, given a node k which is a direct neighbour of nodes i and j, the level of trust node i puts in node j is the product of the trust values of node i in node

k and that of node k in node j. Trust sharing models such as this assume some transitivity of trust, but put stronger emphasis on information obtained from direct neighbours while attenuating trust values received via a multi-hop route. That is, local information carries a greater weight yet can still contribute to a global trust-sharing model.

In [6], the authors take a different approach to sharing trust values by considering trust as an opinion composed of a pair of real numbers (*trustvalue*, *confidence*) $\in [0,1] \times [0,1]$. While *trustvalue* is the estimation of the trustworthiness that node i puts in node j, *confidence* is the accuracy of the *trust value* assignment. In other words, *confidence* can be viewed as the quality of the estimation of trust. Therefore, when a node synthesizes opinions about a distant node, it must take into consideration the confidence value of each local *trustvalue*. We believe that this approach can give a more objective result of trust estimation than in [5] because it recognizes the subjectivity of each local trust estimation. This type of approach would lend itself well to a trust model that used a fuzzy logic reasoning engine.

In this paper, we consider the problem of trust evaluation and show how the quality of trust estimation can be quantified. To start, we state our definition of trust in the next Section.

3 Definition of Trust

Many existing definitions of trust are derivatives of authentication techniques which require encryption and a centralized authentication server. While authentication can provide a quick and efficient way to identify a node, implementing a practical and efficient authentication algorithm in an ad hoc environment remains an open problem [7]. We wish to decouple aspects of trust from authentication so that we may create an additional factor to be used as a tool in securing MANETs. One of the advantages of having a quantifiable and continuous value of trust available is that it allows flexibility in making certain security decisions so that trade-offs between security and functionality can be taken into consideration. We will return to this concept in Section 5.

We define the notion of trust of a given node in a MANET as the consistency of the node's behaviour. The behaviour is observed by other nodes in what is known as a watchdog approach [8]. A consequence of the watchdog approach is that it is observer-dependent; that is, an observed node can have different observational outcomes from the perspective of different neighbours. Let us define the *capacity* of a node to be the innate properties of that node. The node's behaviour will then be inherently tied to, and should reflect, its capacity. If a node behaves inconsistently, it is either because the node is being unfaithful to its capacity, in which case it is acting in an untrustworthy manner, or external factors (e.g. multipath and fading) are affecting its performance. In the latter case, we will assume that the observing nodes are also monitoring the environment and the link quality, can detect such factors, and compensate for them when making their observations.

Some examples of a node's behaviour related to security could be:

– The node's ability to reliably transmit periodic status updates that reflects parameters such as battery level, location and configuration.
– Forwarding packets in a timely manner based on management information base (MIB) bounded delay.
– The difference in the amount of data that should be forwarded and that is actually forwarded.

Our working definition of trust is meant to extract all aspects related to the capacity (as we have defined it) of a node from previous trust models. We do this for the purpose of allowing a quantitative measure of trust to be made which can then be used for making analytical decisions that affect network security. With this new definition of trust, nodes can proactively measure the trustworthiness of their neighbours through observation, without the need of challenging them. Moreover, the network does not need any centralized authentication server to assert signatures.

This definition of trust also allows us to decouple a node's capacity and its trustworthiness. For example, a node may have a large response delay because the throughput aspect of its capacity is low, but this node can still be trusted by other nodes as long as its response delays remain consistently large. As a result, for certain tasks nodes in the network could be selected as a function of both their trustworthiness and their capacities, e.g.: selecting highly capable nodes among those who are above a specified trust threshold.

On the assumption that nodes can monitor the behaviour of their neighbours using a variety of metrics, we will, for the rest of this paper, denote the outcome of a behaviour observation by X, a continuous random variable. X takes values between 0 and 1, thus the outcomes are normalized in the entire network. X is obtained by direct observation by a node i on a node j's behaviour. Values of X can be propagated to the other nodes in the network who can use them as they see fit. Therefore, a given node k can obtain many observation results of a distant node i from different sources (or observers).

The above assumption demands that the observers accurately report all observation results and the use of some cryptography mechanisms prevents the observation results from being modified while they are propagated in the network. The first assumption requires *objectivity* and the second *trust propagation.* These are strong assumptions and it is well recognized that both issues are themselves complex problems in MANET security that need to be addressed separately.

4 Accuracy of Trust Evaluation

In this Section, we are interested in trust as a measure of the consistency of a node's behaviour as objectively observed by one or many different observers.

4.1 Estimation of a Node's Capacities

Let $X_1, X_2, \dots, X_n$ be different observation outcomes of a node i reported by different sources to a node j. If node j computes a weighted average of these values to estimate the capacity of i, then our main concern is how accurate this estimation would be when compared to node i's true capacity. We can subsequently quantify the number of observations needed by node j in order to achieve an acceptable level of confidence in its estimation of node i's capacity. Node j can then decide whether it should trust node i to have such an estimated capacity.

Let $Y = c_1X_1 + c_2X_2 + \ldots + c_nX_n$ be an estimation of node i's capacity, with $0 \leqslant c_i \leqslant 1$ and $\sum_{i=1}^{n} c_i = 1$. We introduce the weights c_i to allow node j to assign different importance to the values X_i, for example: recent observations are more important than old ones. All random variables X_i are assumed to be independent and identically distributed.

Since X measures the observational outcomes of a node's behaviour, $\mu = E[X]$ represents the capacity of this node. Our problem can be formulated as follows: given $\delta \geqslant 0$ and $\epsilon \geqslant 0$, what is the probability that an estimation Y of μ can achieve an accuracy of δ, and conversely, how many observations (n) are needed in order to achieve an estimation of accuracy δ with the probability $1 - \epsilon$?

Theorem 1 (Chernoff bound). *Let* $\mu = E[X]$. *Denote by* $\phi_X(s) = \int_{-\infty}^{\infty} \mathrm{e}^{sx} f_X(x)dx$ *the moment generating function of* X. *We have*

$$P\left[|Y-\mu| \geqslant \delta\right] \leqslant \min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta+\mu)}\prod_{i=1}^{n}\phi_X(c_i s)\right) + \min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta-\mu)}\prod_{i=1}^{n}\phi_X(-c_i s)\right)$$

Proof. We have

$$P\left[|Y-\mu| \geqslant \delta\right] = P\left[Y-\mu \geqslant \delta\right] + P\left[Y-\mu \leqslant -\delta\right].$$

Apply Chernoff bound to random variable Y in the first term yields

$$P\left[Y-\mu \geqslant \delta\right] = P\left[Y \geqslant \delta+\mu\right] \leqslant \min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta+\mu)}\phi_Y(s)\right).$$

Since X_i are independent and identically distributed random variables:

$$\phi_Y(s) = \phi_{\sum_{i=1}^{n} c_iX_i}(s) = \prod_{i=1}^{n}\phi_{c_iX_i}(s) = \prod_{i=1}^{n}\phi_X(c_i s).$$

And hence

$$P\left[Y-\mu \geqslant \delta\right] \leqslant \min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta+\mu)}\prod_{i=1}^{n}\phi_X(c_i s)\right).$$

Similarly, applying the Chernoff bound to the random variable $Z = -Y$ in the second term yields

$$P\left[Y-\mu \leqslant -\delta\right] \leqslant \min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta-\mu)}\prod_{i=1}^{n}\phi_X(-c_i s)\right)$$

which ends the proof. □

If X is a gaussian random variable with mean μ and variance σ^2, then we have the following result.

Corollary 1. *Let* $\xi = \sum_{i=1}^{n} c_i^2$. *If* $X \sim \mathrm{G}(\mu, \sigma^2)$, *then*

$$P\left[|Y - \mu| \geqslant \delta\right] \leqslant 2\exp\left(-\frac{\delta^2}{2\sigma^2\xi}\right)$$

Proof. If $X \sim \mathrm{G}(\mu, \sigma^2)$ then $\phi_X(s) = \exp\left(\mu s + \frac{\sigma^2 s^2}{2}\right)$.

Therefore

$$\begin{aligned}\prod_{i=1}^{n}\phi_X(c_i s) &= \prod_{i=1}^{n}\exp\left(\mu c_i s + \frac{\sigma^2 c_i^2 s^2}{2}\right)\\ &= \exp\left(\mu\textstyle\sum_{i=1}^{n} c_i s + \frac{\sigma^2\sum_{i=1}^{n} c_i^2 s^2}{2}\right)\\ &= \exp\left(\mu s + \frac{\sigma^2\xi s^2}{2}\right).\end{aligned}$$

And hence

$$\min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta+\mu)}\prod_{i=1}^{n}\phi_X(c_i s)\right) = \min_{s\geqslant 0}\left(\exp\left(-\delta s + \frac{\sigma^2\xi s^2}{2}\right)\right).$$

The expression $-\delta s + \frac{\sigma^2\xi s^2}{2}$ has a minimum value of $-\frac{\delta^2}{2\sigma^2\xi}$ when $s = \frac{\delta}{\sigma^2\xi} \geqslant 0$.

Thus

$$\min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta+\mu)}\prod_{i=1}^{n}\phi_X(c_i s)\right) = \exp\left(-\frac{\delta^2}{2\sigma^2\xi}\right).$$

Similar calculations give

$$\min_{s\geqslant 0}\left(\mathrm{e}^{-s(\delta-\mu)}\prod_{i=1}^{n}\phi_X(-c_i s)\right) = \exp\left(-\frac{\delta^2}{2\sigma^2\xi}\right).$$

The assertion thus follows from Theorem 1. □

An interesting conclusion we can draw from Corollary 1 is that the accuracy of the Y-estimation does not depend on the true capacity μ of the subject node. That is, in other words, if node j has received n observations $X_1, \ldots, X_n$ of node i, then it can estimate the capacity of node i with a certain degree of confidence, even if this estimation indicates that node i's capacity is low. Conversely, when some early-arriving reports indicate that node i's capacity is rather high, node j should not rely entirely on this small number of observations to conclude this with a high degree of confidence. Since this result shows that the capacity μ of a node and the estimation accuracy (represented by δ) are statistically unrelated, we do not need to have *a priori* knowledge of a node's capacity in order to draw conclusions about its trustworthiness.

4.2 Number of Observations Required

The following corollary gives a lower bound on the number of observations needed in order to achieve a desired degree of confidence in the estimation.

Corollary 2. *Suppose that* $X \sim \mathrm{G}(\mu, \sigma^2)$*. Given* $0 < \epsilon < 1$*, the minimum number of observations needed in order to achieve an estimation of accuracy* δ *with the probability* $1 - \epsilon$ *is*

$$n \geqslant \frac{2\sigma^2}{\delta^2} \ln\left(\frac{2}{\epsilon}\right)$$

Proof. The inequality in Corollary 1 can be rewritten as

$$P\left[|Y-\mu| \geqslant \delta\right] \leqslant \min_{\xi}\left(2\exp\left(-\frac{\delta^2}{2\sigma^2\xi}\right)\right)$$

where $\xi = \sum_{i=1}^{n} c_i^2 \geqslant \frac{1}{n}$ by Cauchy-Schwartz inequality. Equality occurs when $c_1 = \ldots = c_n = \frac{1}{n}$.

Therefore

$$P\left[|Y-\mu| \geqslant \delta\right] \leqslant 2\exp\left(-\frac{\delta^2 n}{2\sigma^2}\right).$$

And hence

$$P\left[|Y-\mu| < \delta\right] \geqslant 1 - 2\exp\left(-\frac{\delta^2 n}{2\sigma^2}\right) \geqslant 1 - \epsilon$$

yields the desired result. □

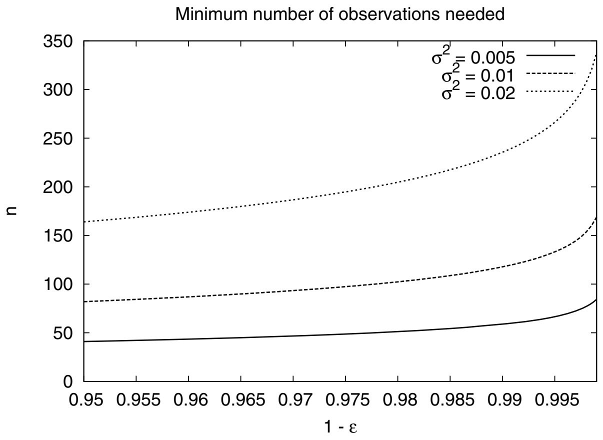


Fig. 1. Minimum number of observations needed to satisfy probability $1 - \epsilon$, with $\delta = 0.03$

Figure 1 shows the minimum number of observations needed in order to achieve an estimation having an accuracy of ± 0.03 ($\delta = 0.03$) with probability at least 95%, for three different values of the variance, σ^2.

Corollary 2 shows that the minimum number of observations required is proportional to the maliciousness of a node which is represented by the standard deviation σ. The deviation measures the lack of consistency in a node's behaviour and we assume intentionally inconsistent behaviour is malicious or untrustworthy. The more inconsistently a node behaves, the more observations we need in order to accurately estimate its capacity. This implies that a greater number of observations are required in order to identify untrustworthy nodes, a fact that makes doing so a more onerous task.

Corollary 2 also gives a lower bound on the number of observations needed to perform an accurate estimation. This lower bound is obtained when equal weights are assigned to each of the observations. Equal weighting of observations is an unlikely scenario: it is more likely a node will grant more importance to recent observations than to stale ones, or more importance to its own observations than to the ones reported by the other nodes. Therefore, the minimum number of observations needed will be higher but will still be accurately quantifiable.

An additional parameter of interest that can be extracted from our calculations is the number of observations required to achieve different accuracies in trust assessment of a node. Figure 2 shows the probability of having an estimation achieve an accuracy δ as a function of the number of observations for a given variance. This figure shows that achieving a very high level of confidence in an assessment comes at great cost with respect to the number of observations required. In this example, it only requires approximately twenty observations to achieve an accuracy of within 4% ($\delta = 0.04$) with 80% confidence but doubling the accuracy to 2% at the same confidence level increases the number of required observations by a factor of six, to $n = 120$. Again, a goal of this work is to allow

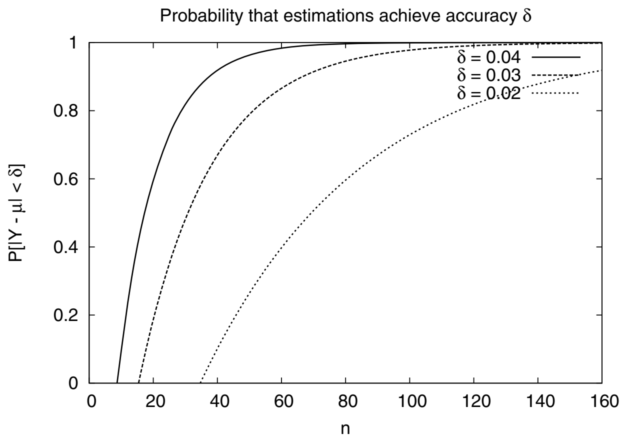


Fig. 2. Probability that the estimations achieve accuracy δ as a function of n, with $\sigma^2 = 0.01$

these tradeoffs to be understood so that decisions using trust as a parameter can be weighed appropriately.

5 Trust and Network Connectivity

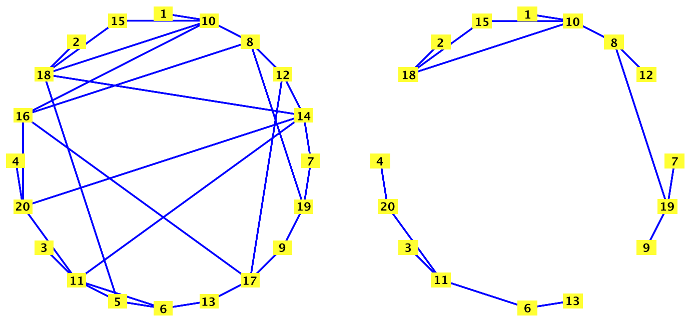
In this section, we study an example where decisions based on trust may have an effect on the connectivity of the network. In particular, we are interested in the probability of the network graph remaining connected if some nodes of the network are untrusted and thus either do not, or are not permitted to, participate in routing. This probability of connectivity is useful for the configuration of trust-based routing. Indeed, when a node extracts a trust topology out of the network graph by excluding nodes having insufficient trustworthiness, it may obtain a partitioned graph and hence trust-based routing may not be available to all destinations.

5.1 Connectivity of Trust-Based Networks

Network connectivity is an important issue in networking and distributed systems. Research on this topic ranges from graph theory [9,10,12] to physics-related domains such as percolation theory [11]. In [12], the authors show that if n nodes of a network are placed uniformly and independently in a unit disc, then the network is connected with a probability asymptotically tending to 1 if and only if each node has $\log n + c(n)$ neighbours and $c(n) \to \infty$ as $n \to \infty$.

In this section, we consider a connected random graph G characterized by n nodes and average density d (or number of neighbours per node). We derive an upper-bound of the connectivity probability for this graph when a subset S, $0 \leqslant |S| \leqslant n$, of randomly chosen nodes in G is untrusted and removed from G.

Figure 3(a) shows an example of a random graph having 20 nodes and average density 3. Nodes {5, 14, 16, 17}, randomly chosen, are untrusted and removed



(a) Initial network graph G

(b) Trust topology, nodes {5,14,16,17} are untrusted and removed from G

Fig. 3. A random graph composed of 20 nodes with 3 neighbours in average per node

from the original graph to obtain a trust topology of the network. This topology is partitioned into two components (see Figure 3(b)).

The following theorem gives us an upper bound of the probability that the trust topology remains connected.

Theorem 2 (Trust-based connectivity).
Let $\rho = \frac{|S|}{n}$, the probability that $G\backslash S$ is connected is

$$P\left[G\backslash S \text{ is connected}\right] \leqslant 1 - \left(\rho\left(2-\rho\right)\right)^{\frac{1}{2}d(1-\rho)}$$

Proof. We prove this theorem by first quantifying the number of edges that are removed from G due to the removal of nodes in S. Then, the remaining induced subgraph $G\backslash S$ is connected if and only if it still has at least one spanning tree.

Since G has n nodes and d neighbours per node on average, the probability that there exists a link between any two nodes is $\frac{d}{n}$. Therefore, the expected number of induced edges of $G\backslash S$, which has $(1-\rho)n$ nodes, is

$$\|G\backslash S\| = \frac{1}{2}\left((1-\rho)n\right)^2 \frac{d}{n} = \|G\|(1-\rho)^2.$$

Hence, the expected number of edges removed from G is

$$\begin{aligned}\|G\| - \|G\backslash S\| &= \|G\|\left(1-(1-\rho)^2\right)\\ &= \|G\|\rho(2-\rho)\end{aligned}$$

which means an edge of G is arbitrarily removed with probability $\rho(2-\rho)$.

Let k be the number of edge-disjoint spanning trees in $G\backslash S$. As each spanning tree in $G\backslash S$ has $(1-\rho)n-1$ edges, we have

$$k \leqslant k_{max} = \frac{\|G\|(1-\rho)^2}{(1-\rho)n-1} \approx \frac{1}{2}d(1-\rho).$$

$G\backslash S$ is disconnected if and only if all its k edge-disjoint spanning trees are disconnected, i.e. at least k edges must be removed from $G\backslash S$ to disconnect it. Hence

$$P\left[G\backslash S \text{ is disconnected}\right] \geqslant \left(\rho(2-\rho)\right)^k \geqslant \left(\rho(2-\rho)\right)^{\frac{1}{2}d(1-\rho)}$$

which ends the proof. □

5.2 Validation

We validate the above analysis by simulation. To start, we fix a value of ρ increasing from 0 to 0.95 by step 0.05, i.e. $\rho = 0, 0.05, 0.1, \ldots, 0.95$. For each value of ρ, we generate 10,000 random graphs. Each graph has 100 nodes and average density $\log_2(100)+1$. Therefore, we ensure that most of the initial graphs are connected (see [12]).

For each random graph G, $\lfloor \rho n \rfloor$ nodes are removed from G. The edges incident to these nodes are also removed. We calculate the percentage of graphs that

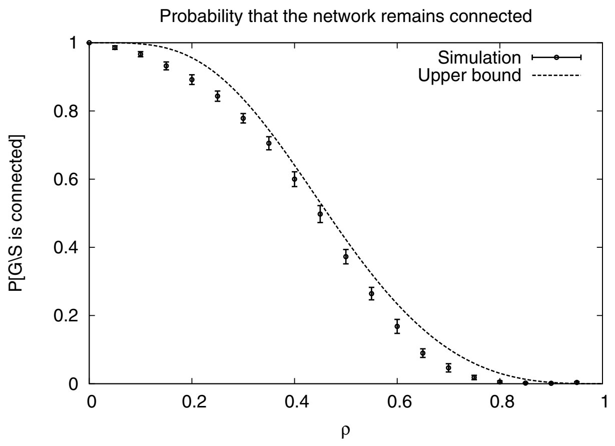


Fig. 4. Probability that a 100-node network ($d = \log_2(100) + 1$) remains connected in presence of a subset S of untrusted nodes, with $|S| = \rho n$

remain connected along with the standard deviation. In total, 200,000 random graphs are generated. The simulations are done using Maple software ([13]).

Figure 4 compares simulation results to analysis. We see that the probabilities of connectivity obtained by simulations closely follow the trend of the upper-bound probabilities obtained by analysis.

The results shown in Fig. 4 demonstrate the potential impact of using strict policies on trust to implement concepts like trust-based routing. If the threshold of required trust is set too high, there is a strong likelihood that a critical number of nodes will be excluded from the network, endangering connectivity. Knowledge of this relationship between trust-level and potential network segregation will allow security decisions to be made in which assuming different levels of risk (routing through less-trusted nodes) can be balanced against the value of increasing the probability of successful message transmission.

6 Conclusion

In this paper, we investigate the issue of trust evaluation and estimation accuracy for ad hoc networks. We start our study by giving a clear definition of trust in the context of ad hoc networks. This definition extracts the physically observable aspects of a nodes behaviour so that each node in the network can decide whether it can trust another node to have certain capacities. We then show that a node's true capacity and its estimation accuracy are statistically independent, given that a node's behaviour follows a normal distribution law. We also provide a minimum number of observations required in order to obtain an accurate estimation of a node's capacity. Given that this minimum number is large, we have shown that an implementation of an analytical trust model will require either a large number of independent observations done in parallel or the ability to cache and safely propagate observation information through the network.

A motivation of this work is to quantify the trade-offs and requirements that will naturally arise by defining trust in this manner for ad hoc networks. To that end, we present an example showing what effect trust-based decisions may have on network connectivity. We derive an upper-bound probability of the network remaining connected when some nodes in that network are untrusted. This information could be used so that trust-based routing is available to as many nodes in the network as possible while simultaneously having an understanding of the measure of risk that is being assumed to do so.

In future work, we can study different mobility scenarios (e.g. time required to compute observations versus speed of nodes) as additional parameters to better understand the tradeoffs and practicability of using trust for security decisions. In addition, an examination of the avenues of attack on this trust model can be considered along with suggestions for mitigating their effects.

Acknowledgement

This work is funded by Defence Research & Development Canada (DRDC).

References

1. Adjih, C., Clausen, T., Jacquet, P., Laouiti, A., Minet, P., Muhlethaler, P., Qayyum, A., Viennot, L.: Optimized Link State Routing Protocol. RFC 3626, IETF (October 2003)
2. Marsh, S.P.: Formalising trust as a computational concept. PhD thesis, University of Stirling (1994)
3. Mui, L., Mohtashemi, M., Halberstadt, A.: A Computational Model of Trust and Reputation. In: Proc. of 35^{th} Hawaii International Conference on System Sciences, HICSS 2002, Hawaii, USA (January 2002)
4. Sun, Y., Yu, W., Han, Z., Ray Liu, K.J.: Trust Modeling and Evaluation in Ad Hoc Networks. In: Proc. of IEEE Global Telecommunications Conference, GLOBECOM'05, St, Louis MO, USA (December 2005)
5. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of Trust and Distrust. In: Proc. of 13^{th} International Conference on World Wide Web, WWW 2004, New York NY, USA (May 2004)
6. Theodorakopoulos, G., Baras, J.S.: Trust Evaluation in Ad-Hoc Networks. In: Proc. of 3^{rd} ACM Workshop on Wireless Security, WiSe 2004, Philadelphia PA, USA (October 2004)
7. Tang, H., Salmanian, M.: Lightweight Integrated Authentication Protocol for Tactical MANETs. In: Proc. of IEEE TrustCom 2008, Zhangjiajie, China (November 2008)
8. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In: Proc. of the Sixth Annual Conference on Mobile Computing and Networking, Boston, MA, USA (August 2000)
9. Gilbert, E.N.: Random Plane Networks. Journal of the Society for Industrial and Applied Mathematics 9(4), 533–543 (1961)

10. Philips, T., Panwar, S., Tantawi, A.: Connectivity Properties of a Packet Radio Network Model. IEEE Transactions on Information Theory 35(5), 1044–1047 (1989)
11. Meester, R., Roy, R.: Continuum Percolation. Cambridge University Press, Cambridge (1996)
12. Gupta, P., Kumar, P.R.: Critical Power for Asymptotic Connectivity in Wireless Networks. In: McEneany, W.M., et al. (eds.) Stochastic Analysis, Control, Optimization and Applications, pp. 547–566. Birkhauser, Boston (1998)
13. Maplesoft: Math Software for Engineers, Educators & Students, `http://www.maplesoft.com`

A Distributed Data Storage Scheme for Sensor Networks

Abhishek Parakh and Subhash Kak

Computer Science Department
Oklahoma State University, Stillwater OK 74075
{parakh,subhashk}@cs.okstate.edu

Abstract. We present a data storage scheme for sensor networks that achieves the targets of encryption and distributed storage simultaneously. We partition the data to be stored into numerous pieces such that at least a specific number of them have to be brought together to recreate the data. The procedure for creation of partitions does not use any encryption key and the pieces are implicitly secure. These pieces are then distributed over random sensors for storage. Capture or malfunction of one or more (less than a threshold number of sensors) does not compromise the data. The scheme provides protection against compromise of data in specific sensors due to physical capture or malfunction.

Keywords: Distributed data storage, sensor networks.

1 Introduction

Sensors may deployed in hostile environment where they may be prone to dangers ranging from environmental hazards to physical capture. Under such circumstances the data and encryption keys stored on these sensors is vulnerable to compromise.

A number of techniques have been proposed for secure communication between sensors using key management and data encryption [1,2,3,4]. Some techniques aim at secure routing [5,6], intrusion detection [7] and others describe a mechanism for moving sensitive data around in the network from time to time [8]. A few researchers propose distributed data storage [15,16,17] but do not adequately address the question of security or use explicit encryption techniques to secure data which leaves the question of secure storage of encryption/decryption keys unanswered.

To go beyond the present approaches, one may incorporate further security within the system by using another layer that increases the space that the intruder must search in order to break a cipher [9,10]. Here, we propose an implicitly secure data partitioning scheme whose security is distributed amongst many sensors. In contrast to hash-based distributed security models for wireless sensor networks [18,19], we consider a more general method of data partitioning. In this approach, stored data is partitioned into two or more pieces and stored at randomly chosen sensors on the network. In scenarios where one or more pieces may be at the danger of being lost or inaccessible due to sensor failure or capture, one may employ schemes that can recreate the data from a subset of original pieces.

A number of schemes have been proposed in the communications context for splitting and sharing of decryption keys [11,20]. These schemes fall under the category of "secret sharing schemes", where the decryption key is considered to be a secret. Motivated by the need to have an analog of the case where several officers must simultaneously use their keys before a bank vault or a safe deposit box can be opened, these schemes do not consider the requirement of data protection for a single party. Further, in any secret sharing scheme it is assumed that the encrypted data is stored in a secure place and that none of it can be compromised without the decryption key.

In this paper, we protect data by distributing its parts over various sensors. The idea of making these partitions is a generalization of the use of 3 or 9 roots of a number in a cubic transformation [12]. The scheme we present is simple and easily implementable.

We would like to stress that the presented scheme is different from Shamir's secret sharing scheme which takes the advantage of polynomial interpolation. Further, Shamir's scheme maps the secret as points on the y-axis,, whereas the scheme proposed in this paper maps the secret as points on the x-axis, as roots of a polynomial.

2 Proposed Data Partitioning Scheme

By the fundamental theorem of algebra, every equation of k^{th} degree has k roots. We use this fact to partition data into k partitions such that each of the partition is stored on a different sensor. No explicit encryption of data is required to secure each partition. The partitions in themselves do not reveal any information and hence are implicitly secure. Only when all the partitions are brought together is the data revealed.

Consider an equation of degree k

$$x^k + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + ... + a_1 x + a_0 = 0 \tag{1}$$

Equation 1 has k roots denoted by $\{r_1, r_2, ..., r_k\} \subseteq \{\text{set of complex numbers}\}$ and can be rewritten as

$$(x - r_1)(x - r_2)...(x - r_k) = 0 \tag{2}$$

In cryptography, it is more convenient to use the finite field $\mathbb{Z}_p$ where p is a large prime. If we replace a_0 in (1) with the data $d \in \mathbb{Z}_p$ that we wish to partition then,

$$x^k + \sum_{i=1}^{k-1} a_{k-i}x^{k-i} + d \equiv 0 \bmod p \tag{3}$$

where $0 \leq a_i \leq p-1$ and $0 \leq d \leq p-1$. (Note that one may alternatively use $-d$ in (3) instead of d.) This may be rewritten as

$$\prod_{i=1}^{k}(x - r_i) \equiv 0 \bmod p \tag{4}$$

where $1 \leq r_i \leq p-1$. The roots, r_i, are the partitions. It is clear that the term d in (3) is independent of variable x and therefore

$$\prod_{i=1}^{k} r_i \equiv d \bmod p \tag{5}$$

If we allow the coefficients in (3) to take values $a_1 = a_2 = ... = a_{k-1} = 0$, then (3) will have k roots only if $GCD(p-1, k) \neq 1$ and $\exists b \in \mathbb{Z}_p$ such that d is the k^{th} power of b. One simple way to chose such a p would be to choose a prime of the form $(k \cdot s + 1)$, where $s \in \mathbb{N}$. However, such a choice would not provide good security because knowledge of the number of roots and one of the partitions would be sufficient to recreate the original data by computing the k^{th} power of that partition. Furthermore, not all values of d will have a k^{th} root and hence one cannot use any arbitrary integer, which would typically be required. Therefore, one of the restrictions on choosing the coefficients is that not all of them are simultaneously zero.

For example, if the data needs to be divided into two parts then an equation of second degree is chosen and the roots computed. If we represent this general equation by

$$x^2 + a_1 x + d \equiv 0 \bmod p \tag{6}$$

then the two roots can be calculated by solving the following equation modulo p,

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4d}}{2} \tag{7}$$

which has a solution in $\mathbb{Z}_p$ only if the square root $\sqrt{a_1^2 - 4d}$ exists modulo p. If the square root does not exist then a different value of a_1 needs to be chosen. We present a practical way of choosing the coefficients below. However, this brings out the second restriction on the coefficients, i.e. they should be so chosen such that a solution to the equation exists in $\mathbb{Z}_p$.

Theorem 1. *If the coefficients a_i, $1 \leq i \leq k-1$ in equation (3) are not all simultaneously zero, are chosen randomly and uniformly from the field, then the knowledge of any $k-1$ roots of the equation, such that equation (4) holds, does not provide any information about the value of d with a probability greater than that of a random guess of $1/p$.*

Proof. Given a specific d, the coefficients in (3) can be chosen to satisfy (4) in $\mathbb{Z}_p$ in the following manner. Choose at randomly and uniformly from the field

$k-1$ random roots $r_1, r_2, ..., r_{k-1}$. Then k^{th} root r_k can be computed by solving the following equation,

$$r_k = d \cdot (r_1 \cdot r_2 \cdot ... \cdot r_{k-1})^{-1} \bmod p \tag{8}$$

Since the roots are randomly chosen from a uniform distribution in $\mathbb{Z}_p$, the probability of guessing r_k without knowing the value of d is $1/p$. Conversely, d cannot be estimated with a probability greater than $1/p$ without knowing the k^{th} root r_k. □

It follows from Theorem 1 that data is represented as a multiple of k numbers in the finite field.

Example 1. Let data $d = 10$, prime $p = 31$, and let $k = 3$. We need to partition the data into three parts for which we will need to use a cubic equation, $x^3 + a_2x^2 + a_1x - d \equiv 0 \bmod p$.

We can find the equation satisfying the required properties using Theorem 1. Assume, $(x-r_1)(x-r_2)(x-r_3) \equiv 0 \bmod 31$. We randomly choose 2 roots from the field, $r_1 = 19$ and $r_2 = 22$. Therefore, $r_3 \equiv d \cdot (r_1 \cdot r_2)^{-1} \equiv 10 \cdot (19 \cdot 22)^{-1} \bmod 31 \equiv 11$. The equation becomes $(x-r_1)(x-r_2)(x-r_3) \equiv x^3 - 21x^2 + x - 10 \equiv 0 \bmod 31$, where the coefficients are $a_1 = 1$ and $a_2 = -21$ and the partitions are 11, 22 and 19.

Choosing the Coefficients: We described above two conditions that must be satisfied by the coefficients. The first condition was that not all the coefficients are simultaneously zero and second that the choice of coefficients should result in an equation with roots in $\mathbb{Z}_p$. Since no generalized method for solving equations of degree higher than 4 exists [21], a numerical method must be used which becomes impractical as the number of partitions grows. An easier method to compute the coefficients is exemplified by Theorem 1 and Example 1.

One might ask why should we want to compute the coefficients if we already have all the roots? We answer this question a little later.

Introducing Redundancy: In situations when the data pieces stored on one or more (less than a threshold number of) sensors over the sensor network may not be accessible, then other sensors should be able to collaborate to recreate the data from the available pieces. The procedure outlined below extends the k partitions to n partitions such that only k of them need to brought together to recreate the data. If $\{r_1, r_2, \ldots, r_k\}$ is the original set of partitions then they can be mapped into a set of n partitions $\{p_1, p_2, \ldots, p_n\}$ by the use of a mapping function based on linear algebra. If we construct n linearly independent equations such that

$$\begin{aligned}
a_{11}r_1 + a_{12}r_2 + \ldots + a_{1k}r_k &= c_1 \\
a_{21}r_1 + a_{22}r_2 + \ldots + a_{2k}r_k &= c_2 \\
&\vdots \\
a_{n1}r_1 + a_{n2}r_2 + \ldots + a_{nk}r_k &= c_n
\end{aligned}$$

where numbers a_{ij} are randomly and uniformly chosen from the finite field $\mathbb{Z}_p$, then the n new partitions are $p_i = \{a_{i1}, a_{i2}, \ldots, a_{ik},\ c_i\}$, $1 \le i \le n$. The above linear equation can be written as matrix operation,

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \vdots & & & \\ a_{n1} & a_{n2} & \ldots & a_{nk} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

To recreate r_j, $1 \le j \le k$ from the new partitions, any k of them can be brought together,

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{bmatrix} = \begin{bmatrix} a_{m1} & a_{m2} & \ldots & a_{mk} \\ a_{n1} & a_{n2} & \ldots & a_{nk} \\ \vdots & & & \\ a_{i1} & a_{i2} & \ldots & a_{ik} \end{bmatrix}^{-1}_{k\times k} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}_{k\times 1}$$

A feature of the presented scheme is that new partitions may be added and deleted without affecting any of the existing partitions.

An alternate approach to partitioning: Once a sensor has computed all the k roots then it may compute the equation resulting from (4) and store one or all of the roots on different sensors and the coefficients on different sensors. Recreation of original data can now be performed in two ways: either using (5) or choosing one of the roots at random and retrieving the coefficients and substituting the appropriate values in the equation (3) to compute

$$-a_0 \equiv x^k + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1 x \bmod p \tag{9}$$

Therefore, the recreated data $d \equiv (p - a_0) \bmod p$. Parenthetically, this may provide a scheme for fault tolerance and partition verification. Additionally, a sensor may store just one of the roots and $k-1$ coefficients on the network. These together represent k partitions.

Note. Distinct sets of coefficients (for a given constant term in the equation) result in distinct sets of roots and vice versa. This is because two distinct sets of coefficients represent distinct polynomials because two polynomials are said to be equal if and only if they have the same coefficients. By the fundamental theorem of algebra, every polynomial has unique set of roots.

Two sets of roots R_1 and R_2 are distinct if and only if $\exists r_i \forall r_j (r_i \ne r_j)$, where $r_i \in R_1$ and $r_j \in R_2$. To compute the corresponding polynomials and the two sets coefficients C_1 and C_2, we perform $\prod_{i=1, r_i \in R_1}^{k} (x - r_i) \equiv 0 \bmod p$ and $\prod_{j=1, r_j \in R_2}^{k} (x - r_j) \equiv 0 \bmod p$ and read the coefficients from the resulting polynomials, respectively. It is clear the at least one of the factors of the two polynomials is distinct because at least one of the roots is distinct; hence the resulting polynomials for a distinct set of roots are distinct.

Theorem 2. *Determining the coefficients of a polynomial of degree $k \ge 2$ in a finite field $\mathbb{Z}_p$, where p is prime, by brute force, requires $\Omega(\lceil \frac{p^{k-1}}{(k-1)!} \rceil)$ computations.*

Proof. If A represents the set of coefficients $A = \{a_1, a_2, ..., a_{k-1}\}$, where $0 \leq a_i \leq p-1$, then by the above note each distinct instance of set A gives rise to a distinct set of roots $R = \{r_1, r_2, ..., r_k\}$, and, conversely, every distinct instance of set R gives rise to a distinct set of coefficients A. Therefore, if we fix d to a constant, then (5) can be used to compute the set of roots. Every distinct set of roots is therefore a $k-1$ combination of a multiset [13,14], where each element has infinite multiplicity, and equivalently a $k-1$ combination of set $S = \{0, 1, 2, ..., p-1\}$ with repetition allowed. Thus, the number of possibilities for the choices of coefficients is given by the following expression

$$\begin{aligned} \left\langle \begin{matrix} p \\ k-1 \end{matrix} \right\rangle &= \binom{p-1+(k-1)}{k-1} \\ &= \tfrac{(p+k-2)!}{(p-1)!(k-1)!} \\ &= \tfrac{(p+k-2)(p+k-3)...(p+k-k)(p-1)!}{(p-1)!(k-1)!} \\ &= \tfrac{(p+k-2)(p+k-3)...p}{(k-1)!} \\ &\geq \lceil \tfrac{p^{k-1}}{(k-1)!} \rceil \end{aligned} \tag{10}$$

Here we have used the fact that in practice $p \gg k \geq 2$, hence the result. We have ignored the one prohibited case of all coefficients being zero, which has no effect on our result. □

3 A Stronger Variation to the Protocol Modulo a Composite Number

An additional layer of security may be added to the implementation by performing computations modulo a composite number $n = p \cdot q$, p and q are primes, and using an encryption exponent to encrypt the data before computing the roots of the equation. In such a variation, knowledge of all the roots and coefficients of the equation will not reveal any information about the data and the adversary will require to know the secret factors of n. For this we can use an equation such as the one below:

$$x^k + \sum_{i=1}^{k-1} a_{k-i} x^{k-i} + d^y \equiv 0 \bmod n \tag{11}$$

where y is a secretly chosen exponent and $GCD(y, n) = 1$. If the coefficients are chosen such that (11) has k roots then

$$\prod_{i=1}^{k} r_i \equiv d^y \bmod n \tag{12}$$

Appropriate coefficients may be chosen in a manner similar to that described in previous sections. It is clear that compromise of all the roots and coefficients will at the most reveal $c = d^y \bmod n$. In order to compute the original value of d, the adversary will require the factors of n which are held secret by the sensor which owns the data.

4 Addressing the Data Partitions

The previous sections consider the security of the proposed scheme when prime p and composite n are public knowledge. However, there is nothing that compels the user to disclose the values of p and n. If we assume that p and n are secret values, then the partitions may be stored in the form of an "encrypted link list", which is a list in which every pointer is in encrypted form and in order to find out which node the present node points to, a party needs to decrypt the pointer which can be done only if certain secret information is known.

If we assume that p and n are public values then the pointer can be so encrypted that each decryption either leads to multiple addresses or depends on the knowledge of the factors or both. Only the legitimate party will know which of the multiple addresses is to be picked.

Alternatively, one may use a random number generator and generate a random sequence of sensor IDs using a secret seed. One way to generate this seed may be to find the hash of the original data and use it to seed the generated sequence and keep the hash secret.

5 Future Work and Other Applications

Our approach leads to interesting research issues such as the optimal way to distribute the data pieces in a network of n sensors. Also in the case when the sensors are moving, one needs to investigate as to how the partitions need to be reallocated so that the original sensor is always able to access the pieces when needed. Further, questions of load balancing so that no one sensor is storing a very large number of partitions and the partitions are as evenly distributed over the network as possible, needs to be investigated.

Yet another application of the presented scheme is in Internet voting protocols. Internet voting is a challenge for cryptography because of its opposite requirements of confidentiality and verifiability. There is the further restriction of "fairness" that the intermediate election results must be kept secret. One of the ways to solve this problem is to use multiple layers of encryption such that the decryption key for each layer is available with a different authority. This obviously leaves open the question as to who is to be entrusted the encrypted votes.

A more effective way to implement fairness would be to avoid encryption keys altogether and divide each cast ballot into k or more pieces such that each authority is given one of the pieces [22]. This solves the problem of entrusting any single authority with all the votes and if any of the authorities (less than the threshold) try to cheat by deleting or modifying some of the cast ballots, then the votes may be recreated using the remaining partitions. Such a system implicitly provides a back-up for the votes.

6 Conclusions

We have introduced a new distributed data storage scheme for the sensor networks. In this scheme data is partitioned in such a way that each partition is implicitly secure and does not need to be encrypted. Reconstruction of the data requires access to a threshold number of sensors that store the data partition.

An additional variation to the scheme where the data partitions need to be brought together in a definite sequence may be devised. One way to accomplish it is by representing partition in the following manner: $p_1, p_1(p_2), p_2(p_3), ...,$ where $p_i(p_j)$ represents the encryption of p_j by means of p_i. Such a scheme will increase the complexity of the brute-force decryption task for an adversary.

References

1. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.: Spins: security protocols for sensor networks. In: Proceedings of ACM Mobile Computing and Networking (Mobicom 2001), pp. 189–199 (2001)
2. Eschenauer, L., Gligor, V.: A Key-management Scheme for Distributed Sensor Networks. In: The 9th ACM Conference on Computer and Communications Security, pp. 41–47 (2002)
3. Liu, D., Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. In: The 10th ACM Conference on Computer and Communcations Security, pp. 52–61 (2003)
4. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In: 10th ACM conference on Computer and Communications Security, pp. 62–72 (2003)
5. Karlof, C., Wagner, D.: Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. In: 1st IEEE International Workshop Sensor Network Protocols and Applications, pp. 113–127 (2003)
6. Yin, C., Huang, S., Su, P., Gao, C.: Secure routing for large-scale wireless sensor networks. Communication Technology Proceedings 2, 1282–1286 (2003)
7. Demirkol, I., Alagoz, F., Delic, H., Ersoy, C.: Wireless Sensor Networks for Intrusion Detection: Packet Traffic Modeling. IEEE Communications Letters 10(1), 22–24 (2006)
8. Benson, Z., Freiling, F., Cholewinski, P.: Simple Evasive Data Storage in Sensor Networks. In: 17th IASTED International Conference on Parallel and Distributed Computing and Systems: First International Workshop on Distributed Algorithms and Applications for Wireless and Mobile Systems, pp. 779–784 (2005)
9. Kak, S.: On the method of puzzles for key distribution. International Journal of Computer and Information Science 14, 103–109 (1984)
10. Kak, S.: Exponentiation modulo a polynomial for data security. International Journal of Computer and Information Science 13, 337–346 (1983)
11. Shamir, A.: How to share a secret. Communication of ACM 22(11), 612–613 (1979)
12. Kak, S.: A cubic public-key transformation. Circuits, Systems and Signal Processing 26, 353–359 (2007)
13. Dickson, L.: Linear Groups with an Exposition of the Galois Field Theory. Dover Publications (1958)
14. Rosen, K.H.: Discrete Mathematics and its Applications. McGraw-Hill, New York (2007)

15. Greenstein, B., Estrin, D., Govindan, R., Ratnasamy, S., Shenker, S.: DIFS: A Distributed Index for Features in Sensor Networks. Ad Hoc Networks 1, 333–349 (2003)
16. Subramanian, N., Yang, C., Zhang, W.: Securing distributed data storage and retrieval in sensor networks. Pervasive and Mobile Computing 3(6), 659–676 (2007)
17. Wang, G., Zhang, W., Cao, G., La Porta, T.: On supporting distributed collaboration in sensor networks. In: IEEE Military Communications Conference, vol. 2, pp. 752–757 (2003)
18. Ye, F., Luo, H., Cheng, J., Lu, S., Zhang, L.: A two-tier data dissemination model for large-scale wireless sensor networks. In: ACM International Conference on Mobile Computing and Networking, pp. 148–159 (2002)
19. Zhang, W., Cao, G., La Porta, T.: Data dissemination with ring-based index for sensor networks. In: IEEE International Conference on Network Protocol, pp. 305–314 (2003)
20. Renvall, A., Ding, C.: A nonlinear secret sharing scheme. In: Pieprzyk, J.P., Seberry, J. (eds.) ACISP 1996. LNCS, vol. 1172, pp. 56–66. Springer, Heidelberg (1996)
21. Bharucha-Reid, A.T., Sambandham, M.: Random Polynomials. Academic Press, New York (1986)
22. Parakh, A., Kak, S.: Internet Voting Protocol Based on Implicit Data Security. In: Proceedings of 17th International Conference on Computer Communications and Networks, ICCCN 2008, pp. 1–4 (2008)

A Rich Client-Server Based Framework for Convenient Security and Management of Mobile Applications

Stephen Badan[1], Julien Probst[1], Markus Jaton[1], Damien Vionnet[2], Jean-Frédéric Wagen[2], and Gérald Litzistorf[3]

[1] University of Applied Sciences of Western Switzerland - HES-SO
HES-SO / HEIG-VD, 1400 Yverdon, Switzerland
markus.jaton@heig-vd.ch
[2] HES-SO / EIA-FR, 1705 Fribourg, Switzerland
jean-frederic.wagen@hefr.ch
[3] HES-SO / HES-GE, 1202 Genève, Switzerland
gerald.litzistorf@hesge.ch

Abstract. Contact lists, Emails, SMS or custom applications on a professional smartphone could hold very confidential or sensitive information. What could happen in case of theft or accidental loss of such devices? Such events could be detected by the separation between the smartphone and a Bluetooth companion device. This event should typically block the applications and delete personal and sensitive data. Here, a solution is proposed based on a secured framework application running on the mobile phone as a rich client connected to a security server. The framework offers strong and customizable authentication and secured connectivity. A security server manages all security issues. User applications are then loaded via the framework. User data can be secured, synchronized, pushed or pulled via the framework. This contribution proposes a convenient although secured environment based on a client-server architecture using external authentications. Several features of the proposed system are exposed and a practical demonstrator is described.

Keywords: mobile security, smartphone, rich client, client-server, secure framework, authentication, transient authentication, theft detection.

1 Introduction

Smartphones are increasingly popular : their performance and recent enhancements in user interface technologies (such as touchscreen, intuitive behavior, ease of use) have popularized their use. Becoming indispensable companion devices, they also keep sensitive information. This covers personal information, such as contacts, addresses, phone calls, to do lists and other data. Usually there are also business data stored on the mobile phone, such as the professional contacts of a bank advisor, or a confidential list of customers data for example. The increasingly popular email clients (whether push or pull) result in the storage of confidential messages. The loss of a mobile phone is not usually considered too critical. Most mobile devices feature easy back-up possibilities on standard personal computers; but the data contained in a lost or stolen smartphone is usually left unsecured, and might be examined by unauthorized or even malicious persons.

Many authors (e.g., [1], [2], [3]) and at least one company, Research In Motion (RIM) [4], have recognized the potential security threat of such devices and proposed interesting solutions. Focusing on readily available solutions, RIM proposes Blackberry. Blackberry devices are very popular in business areas, as the information kept on the device is ciphered and is usually impossible to exploit without proper authentication of the user. Thus, a lost Blackberry device will usually not be readable. The Blackberry solution is however criticized by some potential users, i.e. bank professionals and politics, especially. The problem with the Blackberry solution is mainly the existence within the network of proprietary Network Operations Centers (so-called NOC's) whose functionality is widely opaque to the users. Although the data is fully protected throughout the whole network, the implementation of the security features relies widely on proprietary code located in the NOCs and in the Blackberry phone. This is considered as a possible backdoor by professional users [5].

This contribution proposes an alternative architecture for the implementation of a highly secured client-server infrastructure for mobile applications. The proposed system is called here OSMOSYS (Open and Secure Mobile SYStem). This system features:

– An end-to-end security scheme without opaque components. The code can be independently audited and validated by the entity using the system.
– A convenient management of users, user groups and mobile devices is performed centrally, via a security server.
– Application validations and updates performed remotely on the mobile client.
– Introduction of a so-called companion device which insures periodic user authentication without the need of any manipulation from the user and, thus, offers a protection against loss or theft.
– An advanced development environment allowing the creation of fully secured custom applications.
– A set of useful applications for the secured management of personal data, like e-mail clients, contact list, meetings and notes managers.
– The choice of various devices from several smartphone manufacturers (HTC, LG, SAMSUNG, SONY ERICSSON, ...).

The next section will briefly review the state of the art of some existing solutions to offer security to smartphone users. The requirements for a framework securing user applications are then discussed in Section 3. The main authentication and security features are also explained. The chosen client-server architecture of the framework is then detailed in Section 4. An implementation is described in Section 5 to field test the OSMOSYS framework. Finally, this contribution is concluded in Section 6.

2 State of the Art

As stated in the introduction, many solutions to secure data on mobile devices have been proposed. For example, in [1], a wearable token is used to protect a mobile device: this model was called "transient authentication". The solution requires the author's cryptographic file system (named ZIA). Securing applications on mobile devices also received a lot of attention, including from [2] who designed a flexible code security architecture based on "security-by-contract". Unlike [3], these works do not focus on the needs from the security managers of a fleet of mobile devices.

On the commercial side, one of the leader is actually RIM (Research In Motion) with its BlackBerry devices and associated network [4]. Security on those devices relies mainly on strong ciphering (AES), and optionally on a wireless companion device that can be paired with the mobile to detect loss or theft of the smartphone. The main drawback of the Blackberry security solutions resides in their opacity. The hardware used is proprietary: BlackBerry or some Nokia's compatible devices. The support network relies on a small number of Network Operations Centers (NOCs) whose implementation is largely opaque [4].

Another relatively strong contender is Microsoft in its latest Windows Mobile release 6 (WM6 for short) [6]. WM 6 builds on top of the .NET framework to integrate end-to-end security for mobile applications [7]. WM6, in its version 6.1 in particular, integrates security features that make such applications nearly as secure as their BlackBerry counterparts. On the other hand, the .NET framework code, like the Windows Mobile OS, is not open, and cannot be audited by customers. This leaves place for potential backdoors as in the BlackBerry solution.

Another caveat of the WM6 security scheme is its usage of a mobile VPN as secured transport channel : although the VPN is potentially secure by itself, it remains a low level security solution : if somebody can control the mobile device, the VPN might turn itself into an opened door to the corporate network.

It is actually impossible to speak about mobile solutions without mentioning Apple's iPhone which has revolutionized the way people consider their smartphones. The iPhone is clearly a hot piece of technology, but security is actually not on a par with the ease of use [9]. Moreover, the relatively severe restrictions introduced by Apple to the development of applications discourage developers from developing strong security infrastructures on Apple's platform, although it should not be impossible as our recent work tends to show.

Nokia relies entirely on a clever mobile VPN solution for security. The pertinence of a VPN in a mobile environment may however be questioned, as for the WM6's solution. Third party solutions (commercial, or not [8]) allow to cipher the voice calls or the smartphone at the OS level, but this generally involves relatively complex installation procedures on the devices.

Ideally, it is assumed here that a secure application on a mobile device should,

– belong to fleet of devices managed by a security manager,
– be easy to install and should update itself automatically,
– require very few explicit authentication procedures (e.g., password entry),

– be loss and theft-safe,
– expose only the very necessary subset of data required by the users,
– be able to detect if it has been modified or tampered with, and
– keep secure backup copies of valuable data whenever possible on a remote, trusted, location,

To our knowledge, only RIM's BlackBerry offers nearly such functionalities. But if the user or an IT manager needs to audit the code of the whole system, there is currently no solution. This situation might explain why smartphones are proscribed in many governments, banks and companies using very sensitive information.

3 A Framework for Secured Applications

The main goals of the proposed framework is to authenticate the user and the server as well as to secure the data on the mobile phone in the most convenient manner. Indeed, users will not use or disable annoying security features [1].

Strong authentication methods must be conceived and implemented to access the secured applications and their relevant data. Any remote access to or from the application must use bi-lateral authenticated sessions and ciphered transmissions. Any data on the mobile phone must be either securely deleted when the application closes or ciphered when permanently stored on the mobile device.

Theft or loss protection must be insured: in both cases, the applications and their data should become unusable. A way must also be found to protect the applications so they cannot be copied, modified and re-installed to jeopardize the security.

Thus, the framework should offer convenient solutions to the following issues:

– deployment and control of the life cycle of the applications (installation, upgrade, removal),
– configurations of the security parameters,
– user and user group management, and
– management of the required devices: mobile phones and companion devices.

To determine the required functionalities, a brief scenario is imagined in the next sub-section. A use case follows to define the actors and to detail the authentication services.

3.1 A Sample Scenario

Imagine a key account manager, Alice, having to deal with confidential data related to her customers. Starting her workday, she turns on her smartphone and a so-called companion device small and thin enough to be weared or to fit in her pocket.

On her smartphone, she launches the application called OSMOSYS. This application opens and requests her username and password. Behind the scene, OSMOSYS is also verifying that the companion device can be reached via Bluetooth, and that the smartphone itself is allowed to access the company network.

A set of user applications is then presented to her: an agenda, a to-do list, and a contact list. She opens up her agenda. Nothing has changed since yesterday. But she must prepare herself for an afternoon meeting. Working at her temporary working place in the customer premises, she must go make a few photocopies. She leaves her smartphone on the desk. Despite the fact that this could be considered a security risk, this is not a problem, because the OSMOSYS framework detects that the Bluetooth companion is not within 1 or 2 meters, it locks all applications on the mobile phone and displays a lock on the screen. When Alice returns with her photocopies, the lock disappears and she can use her application as usual.

During lunch, a new appointment is pushed on her agenda, Mr. X must see her at 1pm before the meeting and requests a confirmation call. She calls to confirm. But now being late for her appointment, she forgets her phone on the table at the cafeteria. Fortunately, a colleague brings it to her at Mr. Xs office.

When she tries to use her mobile phone again, she notices that OSMOSYS closed itself to prevent any unauthorized access. This is because the mobile was no longer in range of the companion. All data on the smartphone have also been wiped out. To restore her contacts and meetings, Alice restarts Osmosys, logs onto the server using her user name and password and all her personal data are then automatically restored on the device.

A moment later, when the meeting takes place, she meets a new VIP to be entered on her contact list. At the end of the meeting, on her way back to the hotel, her mobile phone and purse are stolen from her bag. She realizes this only an hour later in the hotel because a message was waiting for her. The phone has been detected as stolen when it lost its connection with the companion., thus OSMOSYS cannot be restarted anymore. In any case, Her SIM card has also been blocked. She must go the premises of their subsidiary to destroy the companion and pick up a new phone and a new companion. A few manual steps in the presence of the local administrator and she will be able to continue her stay and work.

This scenario is not intended to point out all the requirements of a secure framework. Nor does it mention all possible ways to authenticate a user. Furthermore, theft detections via the help of a Bluetooth companion is only one way, although a very promising one. Furthermore, different users might require different security levels.

The OSMOSYS framework aims to offer a convenient way to manage these levels for each user. However, to ease the work of the administrator, different users will be managed as part of different user groups. Security parameters and settings will be defined for each user group.

The reader can imagine the above scenario from the administrator's view point to identify some requirements on the sever part of the OSMOSYS framework. A secure connectivity to the security server, e.g., via SSL, has been implied in the above scenario. Short loss of connectivity is not an issue, but the user name/password authentication requires in our mind a connection to an authentication sever and the client application must authenticate the server.

OSMOSYS deals with this inconvenience using a so-called "robust session" concept. It is up to the security policy of the company to decide whether or not a session may persist in absence of a server connection, and how long the session is allowed to last without any connection to the company's security server.

The next sub-section defines the use case describing the functionalities required to enable scenarios such as the one above.

3.2 The Use Case

There are two main actors: the end user and the security administrator.

The end user uses the applications after being identified and authenticated. Then, existing applications can be upgraded or new applications can be installed.

The security administrator manages:

- the user and user groups,
- the applications (new installation and upgrade),
- the mobile devices,
- the security parameters for each user groups,
- the authentication companion (typically the Bluetooth companion, but possibly a secure card, a list of numbers or of printed code, etc.),
- the remote wipe of the applications and all relevant data.

Evidently the security manager must also be authenticated to use the security server which is located in the company's intranet. To simplify this authentication, we consider here only a central security server. Conventional security solutions such as a restriction to a physical log-in on the security server (Administration console in Figure 1) can solve the main security issues.

The user authentication can be based on several non-exclusive methods. In the considered framework, three authentication methods have been currently defined:

- formal: based on a user name and associated password; as an option the picture of a "turing number" can be used to deter non-human hacking,
- material: typically based on the unique identifier of the mobile phone (International Mobile Equipment Identity: IMEI),
- external: typically using a Bluetooth companion. However, other examples can be devised from a simple paper list of numbers, to secure card displaying response code (in numeral form or to be read by the camera of the phone).

3.3 The Authentication Services

The use case presented here can easily be extended to combine additional authentication techniques. For example, several PC brands as well as a few smartphones can scan a finger print. This kind of biometric identification could be used in addition to other external authentication. Also, the geographic location (via GPS, cell-ID or other means) can play a role in the settings of the security parameters. For example, within the headquarter area, the periodicity of the formal re-authentication is set to its maximum, e.g., 24 hours. Another example is the use of the network connection: while roaming under a foreign operator, external authentication could be verified periodically every 5 seconds, but reported only every 10 minutes to the security server; if a WiFi connection is used then reporting will be more frequent, e.g., every 10 seconds. Again, these authentications techniques can be inserted within the external authentication methods.

The examples mentioned above are provided to illustrate some possible combinations of the authentication methods. When one authentication fails, the procedure is to protect the OSMOSYS applications and sensitive data. The chosen procedure depends on which authentication failed. The OSMOSYS framework offers a set of pre-defined procedures: for example, after the failure of the IMEI authentication and depending on the security settings, the user is either warned or not, the OSMOSYS framework is closed and a flag is set on the security server.

Since many security managers will request their own procedures, the implementation of the OSMOSYS framework should be as open as possible to be easily customized.

Beyond authentication and theft detection, the security framework has also to deal with possible attacks. These features will be not be detailed here. Various solutions have been proposed [2,9] and might be used in the future. Currently, a pragmatic approach was taken and the OSMOSYS framework focuses on providing strong authentication, detecting theft and securing all parameters so they can hardly be tampered with. The current solution is to store all security parameters only on a security server and to download them via a secured HTTPS connection. The downloaded values are then ciphered when stored on the mobile phone. It is recalled that the offline solution is also using the same ciphering procedure to store the data.

The end user must also be insured that the applications used are to be trusted. Thus, the OSMOSYS framework provides this service as explained in the following section.

4 Architecture of the Plaform

The architecture is based on a conventional client server scheme; this choice is mainly dictated by the convenience to have somewhere a centralized administrative tool that controls the features mentioned in Section 3.2. This naturally speaks for the introduction of a centralized server, which will also grant the isolation of the mobile access network from the corporate network. The overall client-server architecture is shown in Figure 1. The security and applications servers on the servers' side can obviously run on the same physical machine.

To achieve the goal of being able to check the integrity of an application, and to make sure that a modified application could not access valuable data within the corporate network, an obvious solution is to divide every application into server and client stubs. Defining a very strict, although flexible, interface between client and server part of the application insures that a modified application will not be able to access data outside the scope of the application. To further enhance

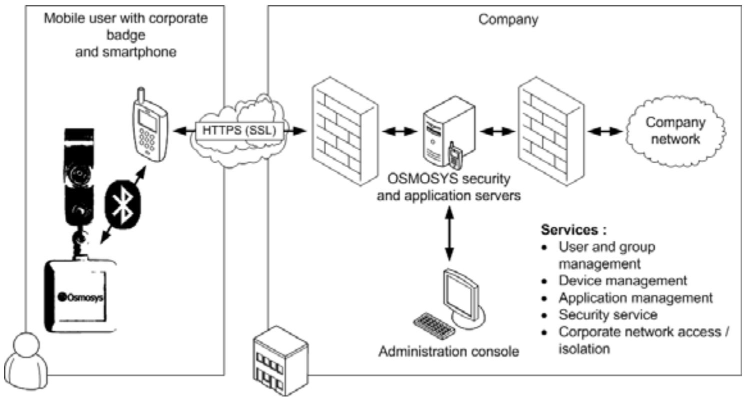

Mobile user with corporate badge and smartphone
Company
HTTPS (SSL)
OSMOSYS security and application servers
Company network
Osmosys
Services :
• User and group management
• Device management
• Application management
• Security service
• Corporate network access / isolation
Administration console

Fig. 1. The client-server architecture of the Open Secured Mobile System (OSMOSYS). A companion device is linked to the client to ease theft or loss detection. The server side is logically composed of a security server and an application server.

security, applications are systematically signed, and authenticated by the server stub. The security and application servers are detailed in Figure 2.

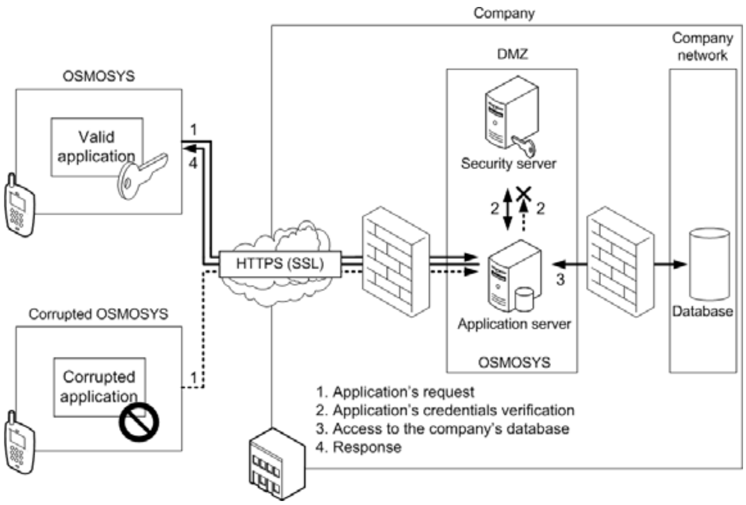


Fig. 2. Details of the client-serveur interaction to access credentials and secure an application: the application server verifies the validity of the requests on the security server before executing them

The client application also checks the server's identity before beginning any transactions; beside this check, it also authenticates the user and the companion device. Authentication is always performed with the assistance of the security server. When the server is unreachable, it will depend on the local security

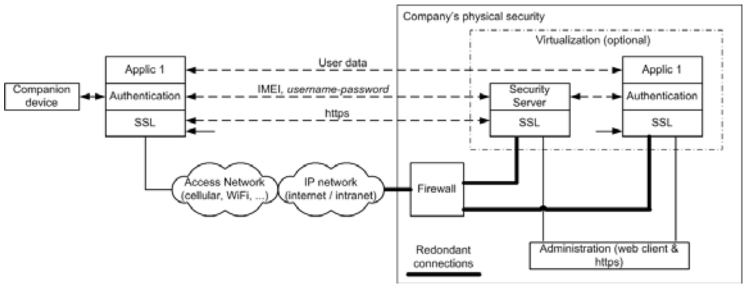

Company's physical security
Virtualization (optional)
User data
IMEI, username-password
https
Companion device
Applic 1
Authentication
SSL
Security Server
SSL
Applic 1
Authentication
SSL
Access Network (cellular, WiFi, ...)
IP network (internet / intranet)
Firewall
Redondant connections
Administration (web client & https)

Fig. 3. Architectural view of the OSMOSY framework from a security point of view

settings wether the protected applications are usable or not. The most stringent conditions will require a permanent secure connection to the security server. In this case, a temporary loss of connection due to radio coverage problems for example, would automatically lock the secured OSMOSYS applications until a connection can be re-established by the client.

4.1 Risk Analysis

To analyze the proposed OSMOSYS architecture against the possible threats, the Figure 3 clarifies the client-server interactions. This section presents a brief evaluation of the risks to the OSMOSYS system.

– Write access to the database of the Security Server: This is probably the greater risk, but the security server will usually be physically protected within the company. The access to the servers is then controlled at the physical and administrative levels.

– Unauthorized user: Strong authentications can be based on using material: IMEI (mobile device), formal: User Name, Password, and external (Bluetooth companion). A mix of these authentication techniques can be adapted to the customer's requirements.

– Server authentication: The security server is authenticated via PKI. Therefore, phishing attacks are mitigated via careful PKI implementation and testing.

– Access rights: Each user is only authorized to use his own set of of applications according to his user group profile.

– Stolen mobile: Sensitive user data (contacts, messages, ...) are ciphered at the application level by the OSMOSYS framework. Furthermore, if the mobile is separated from the Bluetooth companion, the OSMOSYS framework will block, and a moment later close, all secured applications.

– Eavesdropping: The confidentiality of all exchanges (mobile-server and administrator-server) are insured via SSL (AES128). The Bluetooth connection to the companion should always be ciphered.

– Data integrity: SSL provides the data integrity service.

– Denial of service (DoS): On the server side, redundancy links and a restriction to intranet access are used to minimized the risks. On the mobile side, the DoS risks on the access network are considered to be small. The DoS via Bluetooth will currently close the OSMOSYS framework.

– Tracability: Logs are generated on the security server. Logs on the mobile devices were considered only for debugging in order to simplify the mobile phone software and to prevent any security risk.

5 The Open and Secure Mobile System

To validate the concept, a practical system has been set up corresponding to the above specifications. First, an early prototype has been built on a J2ME client, and a JEE server. Although the concept could be effectively validated on this base, J2ME proved to be too restrictive for low-level control, thus a production-ready version has been developed using Microsoft's .NET framework (C# managed code) on the client, but still keeping the JEE environment on the server side.

The OSMOSYS framework is an application, from .NET's point of view, and thus does not rely on Windows Mobile 6 security infrastructure. Instead, it implements its own secured environment at the application level.

5.1 OSMOSYS Client and Companion

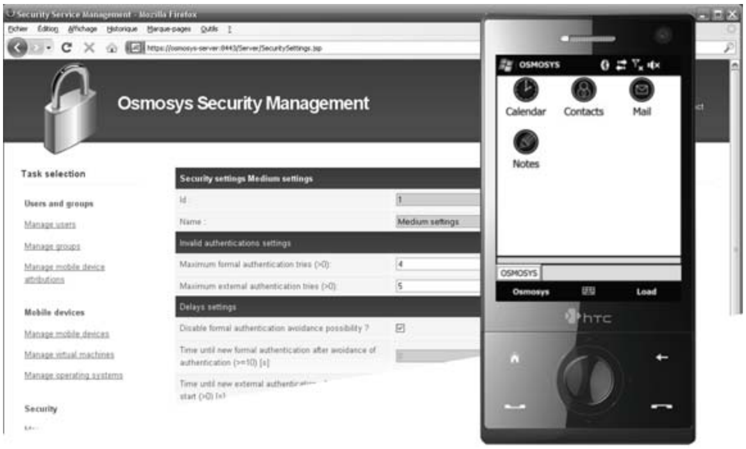
As stated above, the client device is a conventional WM6 smartphone. Installation of OSMOSYS is easy: open Internet Explorer, write the URL of your corporate OSMOSYS server, and click on the installation icon on the displayed page. The network administrator should have first registered a few data such as the IMEI and the Bluetooth address of the companion device. This condition allows to authenticate the smartphone and its Bluetooth companion device from the server side. From this point, OSMOSYS takes all necessary measures to insure that

- the user has the latest OSMOSYS framework version,
- the user can use the applications that are defined for him (his user group),
- the user has the latest software versions of the applications

From now on, the user may use OSMOSYS and the corresponding applications as long as the company's security requirements are met. Figure 4 show the OSMOSYS client on a smartphone after a successful authentication, before any user application has been launched.

5.2 OSMOSYS Server

The server is a conventional JEE application server. The current version has been deployed on a Tomcat server with EasyBeans as EJB container, and the administration dialog is implemented using conventional JSP technology, but this configuration is in no way a must. Any JEE implementation (capable of running servlets and featuring an EJB container) could be used. The server is divided into an application server and a security server; currently both run on the same machine. The role of both servers is well defined:


Security Service Management - Mozilla Firefox
Fichier Edition Affichage Historique Marque-pages Outils ?
https://osmosys-server:8443/Server/SecuritySettings.jsp
Osmosys Security Management
Task selection
Users and groups
Manage users
Manage groups
Manage mobile device attributions
Mobile devices
Manage mobile devices
Manage virtual machines
Manage operating systems
Security
OSMOSYS
Calendar
Contacts
Mail
Notes
OSMOSYS
Osmosys
Load
htc

Security settings Medium settings

Id : 1

Name : Medium settings

Invalid authentications settings

Maximum formal authentication tries (>0): 4

Maximum external authentication tries (>0): 5

Delays settings

Disable formal authentication avoidance possibility ? ☑

Time until new formal authentication after avoidance of authentication (>=10) [s]

Time until new external authentic start (>0) [s]

Fig. 4. Screen shots from the OSMOSYS server (behind) and client (mobile in front)

– The security server deals with authentication and security parameters concerning the users, user groups and devices (Figure 4),
– The application server deals with the server-side of the applications.

Any application built on top of OSMOSYS automatically inherits the security mechanisms built into the framework; if a new application has to be developed, it will be separated into a client part and a server part that share a common communication mechanism. All security and communication issues are kept hidden by the framework : the applications have no way to short-circuit the framework, so they are commited to use OSMOSYS's rules defined by the framework.

As every application has a client and a server part, it is very unlikely that a client could be modified and still work with the server counterpart, provided its signature could have been kept unchanged or faked by some way.

Special care has been taken to allow the full control of the whole system from the server's administration console. It is very unlikely that an end user has to install anything on the smartphone before using OSMOSYS. It is however possible that for security reasons, some companies would require to uninstall some or all pre-installed applications. It remains to be investigated whether or not this uninstallation can be performed remotely from the security server.

6 Conclusion and Future Work

Some important requirements to secure applications and sensitive data on smartphones have been presented. A rich client-server architecture was proposed to authenticate the user, detect loss or theft with the help of an additional companion device, and to secure the use of end-user applications. The security features include strong authentication, data ciphering, and management of the life cycle for any customized application. The framework named OSMOSYS was described to offer these various security services.

An implementation of OSMOSYS using Windows Mobile 6 demonstrated the feasibility of our approach. The security server has been audited by the last author and no flaw has been found. The OSMOSYS product is currently available as a public service for evaluation and testing (see www.securemesa.ch). At the time of writing, a private bank in Geneva is evaluating the service under realistic conditions. Preliminary results of this field trial will be presented at the conference.

References

1. Nicholson, A.J., Corner, M.D., Noble, B.D.: Mobile Device Security Using Transient Authentication. IEEE Transactions on Mobile Computing 5(11), 1489–1502 (2006)
2. Desmet, L., Joosen, W., Massacci, F., Naliuka, K., Philippaerts, P., Piessens, F., Vanoverberghe, D.: A flexible security architecture to support third-party applications on mobile devices. In: Proceedings of the 2007 ACM Workshop on Computer Security Architecture. CSAW 2007, Fairfax, Virginia, USA, pp. 19–28. ACM, New York (2007)
3. Wu, H., Grgoire, J., Mrass, E., Fung, C., Haslani, F.: MoTaskit: a personal task-centric tool for service accesses from mobile phones. In: Proceedings of the 1st Workshop on Mobile Middleware: Embracing the Personal Communication Device. MobMid 2008, Leuven, Belgium, pp. 1–5. ACM, New York (2008)
4. The BlackBerry solution allows users to stay connected with wireless access to email, corporate data, phone, web and organizer features, http://www.blackberry.com/
5. Hoffman, D.V.: Blackjacking: Security Threats to BlackBerry Devices, PDAs and Cell Phones in the Enterprise. Wiley, Chichester (2007)
6. Windows Mobile 6, http://msdn.microsoft.com/en-us/library/bb158486.aspx
7. Understanding the Windows Mobile security model, http://technet.microsoft.com/en-us/library/cc512651.aspx
8. Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring integrity on mobile phone systems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies. SACMAT 2008, Estes Park, CO, USA, pp. 155–164. ACM, New York (2008)
9. Dunham, K. (ed.): Mobile Malware Attacks and Defense. Elsevier, Amsterdam (2009)

A Robust Conditional Privacy-Preserving Authentication Protocol in VANET*

Chae Duk Jung[1], Chul Sur[2], Youngho Park[3], and Kyung-Hyune Rhee[3]

[1] Department of Information Security, Pukyong National University, 599-1, Daeyon3-Dong, Nam-Gu, Busan, 608-737, Republic of Korea
jcd0205@pknu.ac.kr

[2] Department of Computer Science, Pukyong National University
kahlil@pknu.ac.kr

[3] Division of Electronic, Computer & Telecommunication Engineering, Pukyong National University
{pyhoya,khrhee}@pknu.ac.kr

Abstract. Recently, Lu *et al.* proposed an efficient conditional privacy preservation protocol, named ECPP, based on group signature scheme for secure vehicular communications. However, ECPP dose not provide unlinkability and traceability when multiple RSUs are compromised. In this paper, we make up for the limitations and propose a robust conditional privacy-preserving authentication protocol without loss of efficiency as compared with ECPP. Furthermore, in our protocol, RSUs can issue multiple anonymous certificates to an OBU to alleviate system overheads for validity check of RSUs. In order to achieve these goals, we consider a universal re-encryption scheme as our building block.

Keywords: Vehicular Ad-hoc Network, Conditional Privacy, Authentication, Movement Tracking, Universal Re-encryption, Group Signature.

1 Introduction

In the near future, vehicles will be equipped with on-board processing and wireless communication modules, which enable vehicle-to-vehicle and vehicle-to-infrastructure communications based on short-range wireless technology, e.g., IEEE 802.11p [16]. That is called a vehicular ad-hoc network (VANET). VANET mainly consists of On-Board Units (OBUs) and Roadside Units (RSUs) [11], where OBUs are installed on vehicles to provide wireless communication capability, while RSUs are deployed to provide access point to vehicles within their radio coverages. By this organization, VANET can provide useful functions such as cooperative driving and probe vehicle data. For example, a vehicle can warn other vehicles about traffic accidents or traffic jam.

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-521-D00454).

Considering the useful VANET applications, it is necessary to develop a suit of elaborate and carefully designed security mechanisms to make VANET applications viable [1,5,7,10,12,13,14,15,17]. Especially, the increasing demand for privacy protection has brought additional requirements related with privacy preservation in VANET such as anonymous message authentication. Therefore, conditional privacy preservation, which private information is not disclosed to other entities while the authorities should legally trace user-related information in case of a disputed event, becomes one of the main requirements for secure VANET.

Lin *et al.* [9] proposed a conditional privacy-preserving authentication protocol for VANET by integrating the techniques of group signature and identity-based signature. However, in [9], even though the authors reduced certificate revocation list (CRL) size, their protocol needs the management of certificate revocation information in OBUs. Hence, each vehicle must spend much time in message verification when the number of revoked vehicles is increased.

Recently, Lu *et al.* [8] proposed an efficient conditional privacy preservation protocol, named ECPP, which issues on-the-fly short-time anonymous certificate to OBUs by using a group signature scheme [2]. Since RSUs can check the validity of the requesting vehicle during the short-time anonymous certificate generation phase, such revocation check by an OBU itself of [9] is not required. ECPP provides unlinkability and traceability under the unrealistic assumption that most RSUs will not disclose any inner information without the authorization of the trusted authority. However, due to the fact that there exist a large number of RSUs, cost considerations prevent the RSUs from having sufficient protection facilities against malicious attacks. Therefore, it is possible for an attacker to access RSUs and disclose the information in the RSUs. When multiple RSUs are compromised, an attacker is able to track the movement trace of a vehicle by using the information stored in the compromised RSUs [3], because each RSU stores unchanged pseudonyms for OBUs in ECPP. As a result, ECPP does not provide unlinkability of OBUs when some RSUs were compromised. Furthermore, since the trace protocol in ECPP is run by incorporating with an RSU which issued a certificate corresponding to a disputed message, it is impossible to trace OBUs belong to compromised RSUs. Consequently, ECPP does not provide unlinkability and traceability when multiple RSUs are compromised.

Moreover, even though ECPP runs mutual authentication between OBUs and RSUs, it requires validity check of RSUs by using up-to-date revocation list in anonymous certificate generation phase by considering an attacker who can disclose inner information in compromised RSUs.

As a result, it is necessary to design an efficient and robust conditional privacy-preserving authentication protocol that not only provides unlinkability and traceability even if multiple RSUs are compromised, but also reduces system overheads for validity check of RSUs in OBUs' anonymous certificate generation phase.

Our Contribution. In this paper, we propose a robust conditional privacy-preserving authentication protocol based on universal re-encryption scheme [4] and identity-based group signature scheme [2]. To efficiently resolve the problem resulted from certificate revocation in traditional PKI, the proposed protocol employs the concept of shot-time anonymous certificate. Besides, even though multiple RSUs are compromised, in contract to ECPP, the proposed protocol provides unlinkability and traceability without loss of efficiency as compared with ECPP. Furthermore, to reduce system overheads for validity checks of RSUs in OBUs' anonymous certificate generation phase, RSUs issue multiple anonymous certificates to an OBU based on universal re-encryption scheme.

The rest of the paper is organized as follows. In Section 2, we describe our design objectives and define conditional privacy level for secure VANET. We outline our system architecture in Section 3. Section 4 presents the proposed conditional privacy-preserving authentication protocol. In Section 5, the security and efficiency of the proposed protocol are analyzed. Finally, we conclude the paper in Section 6.

2 Security Requirements

In this paper, we aim at achieving the following objectives:

- **Authentication.** The origin of the messages should be authenticated to guard against impersonation attack. Also, even though an attacker compromises some RSUs, the attacker cannot forge a signature on safety message in a compromised RSU's communication range.
- **Anonymity.** The identities of vehicles should be hidden from normal message receivers during the safety message authentication process. Moreover, even if an attacker obtains inner information of compromised RSUs, the attacker cannot disclose real identities of OBUs.
- **Unlinkability.** When an adversary has collected several safety messages that have different pseudonyms from an OBU, the OBU should be still not traceable. Moreover, even though the adversary compromised RSUs, it also cannot link information stored in the RSUs as the same OBU.
- **Traceability.** The authority should be able to trace the sender of a safety message by revealing the identity in case of any disputed situation such as liability investigation. In addition, even if multiple RSUs are compromised, the authority should be able to trace real identities of pseudonyms in anonymous certificates without assistance of compromised RSUs.

Following the above goals, we modify the definition of conditional privacy level in [8] as Table 1.

Note that, ECPP does not provide unlinkability of an OBU since the compromised RSUs store the same pseudonym for same OBU. Moreover, since the trace protocol in ECPP is run by the trust authority with an RSU (certificates generator), it is impossible to trace real identities of OBUs belong to damaged RSUs. As a result, ECPP provides Level 1 Privacy in Table 1.

Table 1. Definition of Conditional Privacy Level

	Authentication	Anonymity	Unlinkability	Traceability
Level 1 Privacy	O	O	×	×
Level 2 Privacy	O	O	O	×
Level 3 Privacy	O	O	O	O

3 System Model

In this section, we describe our system model, in which communication nodes are either membership manager, RSUs, or OBUs as follows:

- **Membership Manager** is public agencies or corporations with administrative powers in a specific field; for example, city or state transportation authorities. The membership manager establishes and manages system parameters and system roles for secure VANET. In addition, the membership manager should be able to reveal the identities of safety message senders in the case of disputed traffic events.
- **RSUs** belong to the membership manager. When an RSU receives a request message for certificate issue from an OBU, it checks the validity of the OBU with the membership manager. If the OBU is legal, the RSU issues multiple anonymous certificates to the OBU by using universal re-encryption scheme and group signature scheme.
- **OBUs** periodically send a safety message by using its own short-time anonymous certificate. When an OBU needs anonymous certificates, it requests certificate issue to a nearby RSU. If the OBU is legitimated, it is able to get new multiple short-time anonymous certificates from the RSU.

To make our model more clear, we assume the followings:

- Each OBU has a unique electronic identity, e.g., ELP (Electronic License Plate).
- OBUs change its own shot-time anonymous certificates within 1 minute (min) as a result of [12].
- Membership manager can inspect all RSUs at high level and detect compromised RSUs.

Our protocol consists of the following 5-phases:

1. Setup: The membership manager sets up its own master key and system parameters based on identity-based group signature scheme [2] and universal re-encryption scheme [4]
2. Registration: The membership manager assigns MAC keys to OBUs and group signing keys to RSUs, respectively. At the same time, the membership manger stores the pairs ⟨OBU's real ID, MAC key⟩ in his secure storage.

3. Multiple Anonymous Certificates Generation: When an OBU requests anonymous certificates for given time period, it generates and transmits a request message including new pseudonym and MAC value to a nearby RSU. After validity check for the OBU, the RSU issues multiple short-time anonymous certificates to the OBU. Finally, the OBU verifies the issued certificates and checks the validity of the RSU by using RL (Revocation List).
4. Safety Message Authentication: OBUs periodically sign traffic information by using conventional digital signature scheme under its own shot-time signing key, and then broadcast a traffic information attached with the signature and the short-time anonymous certificate. Before accepting received traffic information, each receiver verifies the signature with sender's certificate.
5. OBU's Real ID Trace: In case of problematic happening, the membership manager traces the real identity of generator of a safety message by using its own private key.

4 Proposed Protocol

In this section, we propose a robust conditional privacy-preserving authentication protocol based on universal re-encryption [4] and identity-based group signature [2] for secure VANET. Table 2 describes the notations used in the proposed protocol.

Table 2. Notations

Notation	Description
GS_{mk}	master key of group signature
GS_{pk}	public key of group signature
GS_{params}	system parameters of group signature
sk_{MM}, pk_{MM}	private/public key pair of membership manager
H, H_0	cryptographic hash functions
K_i	MAC key for OBU_i
$H_K()$	MAC function under key K
ID_i	OBU_i's real identity, i.e., ELP
ID'_i	pseudonym of OBU_i
$ID'_{i,\cdot}$	short-time pseudonym of OBU_i
$Cert_{i,\cdot}$	short-time anonymous certificate of OBU_i
$sk_{i,\cdot}, pk_{i,\cdot}$	OBU_i's short-time private/public key pair
$GSig()$	group signature function
$Sig()$	ordinary digital signature function
$E(), D()$	encryption and decryption function of universal re-encryption
$Re()$	re-encryption function of universal re-encryption
t	validity period for shot-time certificate

4.1 Setup

The membership manager generates the required bilinear groups and system parameters as it is in [2]. Given security parameter k, the membership manager chooses a k-bit prime number p, bilinear map groups (G_1, G_2) of order p. The membership manager randomly picks generators $g_1 \in G_1$ and $g_2 \in G_2$. Let $\hat{e} : G_1 \times G_2 \rightarrow G_T$ be a bilinear pairing. The membership manager selects $\gamma \in Z_p^*$ and sets $GS_{pk} = g_2^{\gamma}$. After that, the membership manager chooses secure cryptographic hash functions $H : \{0,1\}^* \rightarrow Z_p^*$ and $H_0 : \{0,1\}^* \rightarrow G_2^2$. The system parameter GS_{params} and master key GS_{mk} of the membership manager are set up as follows:

$$GS_{params} = (G_1, G_2, G_T, \hat{e}, p, g_1, g_2, H, H_0, GS_{pk}),\ GS_{mk} = \gamma$$

In addition, the membership manager selects its own private key $sk_{MM} \in Z_p^*$ and computes public key $pk_{MM} = g^{sk_{MM}}$, where g is a generator for the underlying group for the ElGamal cryptosystem.

4.2 Registration

In registration phase, the membership manager issues a group signing key to each RSU for generating anonymous certificates and a MAC key to each OBU. All OBUs need to be registered with the trusted membership manager and preloaded with public system parameters and their own MAC key before joining VANET. Thus, the membership manager randomly chooses MAC key K_i and then transmits K_i to OBU_i over a secure channel. In addition, the membership manager stores the pair $\langle ID_i, K_i \rangle$ in its own secure storage. To generate a group signing key, the membership manager selects $x_j \in Z_p^*$ such that $\gamma + x_j \neq 0$, and then sets $A_j = g_1^{1/\gamma + x_j}$. After that, the membership manager sends (A_j, x_j) to RSU_j as the group signing key over a secure channel.

4.3 Multiple Anonymous Certificates Generation

When an OBU_i wants to get new multiple anonymous certificates for given time period from a nearby RSU_j, OBU_i and RSU_j run anonymous certificate generation protocol as follows.

Step 1. An OBU_i encrypts its own real identity ID_i based on universal re-encryption under the membership manager's public key to generate a new pseudonym ID_i'. At the same time, the OBU_i randomly selects a short-time validity period t and multiple signing keys $sk_{i,1}, \ldots, sk_{i,n}$, and then computes corresponding public keys (note that, each public key is formed by signature scheme in safety message authentication phase). Finally, the OBU_i computes MAC value $MAC_{ID_i'} = H_{K_i}(ID_i' || PK || t)$ and transmits $\langle ID_i', PK, t, MAC_{ID_i'} \rangle$ to a nearby RSU_j for obtaining n anonymous certificates, where PK is a set of public keys.

Step 2. The RSU_j checks the valid period t since a long period will cause the risk of continued circulation of an invalid certificate by an attacker. If t is short valid period, RSU_j transmits the received message to the membership manager for checking a validity of OBU_i. The membership manager decrypts the received pseudonym ID'_i for getting real identity ID_i and searches MAC key K_i corresponding to ID_i. If $MAC_{ID'_i}$ is valid and the OBU_i is legal, the membership manager sends the permission message and up-to-date RL (=$\{A_1, \ldots, A_{N_R}\}$, Revocation List for revoked RSUs) to the RSU_j.

Step 3. If the RSU_j receives the permission message from the membership manager, it repeatedly executes a re-encryption algorithm in [4] with the pseudonym ID'_i to obtain n pseudonyms $ID'_{i,1}, \ldots, ID'_{i,n}$ for OBU_i. Then, the RSU_j computes each $\sigma_l = GSig(ID'_{i,l}||pk_{i,l}||t)$ by using group signature scheme [2] under its own group signing key and forms each anonymous certificate $Cert_{i,l} = \{ID'_{i,l}, pk_{i,l}, t, \sigma_l\}$, where $l = 1, \ldots, n$. Finally, the RSU_j transmits multiple anonymous certificates $Cert_{i,1}, \ldots, Cert_{i,n}$ and RL to the OBU_i.

Step 4. The OBU_i checks the validity of the RSU_j by using revocation checking procedure in [2] with RL. If RSU_j is not revoked, the OBU_i verifies received certificates by using GS_{pk}. Finally, the OBU_i accepts issued certificates if all the checks are valid.

4.4 Safety Message Authentication

OBU_i signs a traffic information m by using a conventional digital signature scheme such as ECDSA under its own short-time signing key $sk_{i,l}$ for generating signature σ_m. Then, the OBU_i forms the safety message $Msg = \{m, \sigma_m, Cert_{i,l}\}$ and broadcasts Msg. Upon receiving a safety message, each receiver first checks the validity of $Cert_{i,l}$ by using GS_{pk}. If the $Cert_{i,l}$ is valid, the receiver retrieves $pk_{i,l}$ from the $Cert_{i,l}$ and verifies σ_m using the $pk_{i,l}$. If σ_m is valid, the traffic information can be accepted, otherwise discarded.

4.5 OBU's Real ID Trace

In case of any disputed situation, it is necessary to extract a real identity of generator of the broadcasted safety message by the membership manager. Since pseudonyms $ID'_{i,l} = Re(ID'_i)$ of OBU_i are computed by re-encryption scheme of [4] with $ID'_i = E_{pk_{MM}}(ID_i)$, the pseudonyms were formed as ciphertexts for OBU's real identity under the pubic key of the membership manager. Note that, due to the property of universal re-encryption, the membership manager can output real identities from pseudonyms in the safety message by using its own private key sk_{MM}.

5 Discussion

5.1 Security

We analyze how the proposed protocol satisfies the security requirements stated in Section 2.

- **Authentication.** Since the signature is generated by a conventional digital signature scheme with respect to a pseudonym and a corresponding public key, which was proven to secure against adaptive chosen message attack, no adversary can launch a forgery attack and an impersonation attack to an OBU.
- **Anonymity.** Since OBUs' real identities are encrypted under pk_{MM} and re-encrypted ciphertexts are used as pseudonyms, an attacker who compromised multiple RSUs cannot disclose a real identity from pseudonyms in certificates without knowing sk_{MM} due to the property of universal re-encryption.
- **Unlinkability.** An eavesdropper cannot link the safety messages since most safety message consists of different pseudonyms and public keys independently. Even though multiple RSUs are compromised, the attacker does not obtain any information from the compromised RSUs since each OBU generates and transmits different pseudonyms to RSUs in certificate generation.
- **Traceability.** In dispute cases, the membership manager is able to trace a real identity of OBU_i corresponding pseudonym $ID'_{i,\cdot}$ by using its own private key sk_{MM}. Even if some RSUs are compromised, the membership manager is able to trace OBUs since the trace procedure in the proposed protocol is executed by the membership manager without cooperations with RSUs.

As a result, the proposed protocol provides Level 3 Privacy in Table 1.

5.2 Efficiency

In this section, we compare the proposed protocol with ECPP to show that our protocol provides reasonable efficiency in terms of OBU's computational costs and RSU valid serving ratios. For fairness in comparisons, we selected a bilinear pairing of 80-bit security level as the same security measures of ECPP as follows; degree $k = 6$, $|G_1| = 160$ bits and $|q| = 1024$ bits. Table 3 shows the measures to estimate and to compare our protocol with ECPP. Note that ECPP$_M$ is ECPP's multiple certificates issue type as given in Table 4.

Table 3. Cryptographic operation time and protocol execution time (implemented on Pentium IV 3.0 GHz, $N_R = |RL|$)

Cryptographic operation time	Time(ms, millisecond)	
$\hat{e}$ bilinear pairing operation	4.5	
point multiplication on G_1	0.6	
exponentiation on Z_q	2.1	
Protocol execution time	ECPP$_M$(ms)	Our Protocol(ms)
time for n certificates issue	$20.4+14.4n$	$6.3+18.6n$
time for n certificates verification	$17.1n$	$17.1n$
time for validity check of RSU	$9N_R$	$9N_R$
total time for n certificates generation	$20.4+31.5n+9N_R$	$6.3+35.7n+9N_R$

Table 4. Modification to multiple-ECPP

Phase	ECPP	ECPP$_M$
public key generation	$x \in Z_q^*,\ Y = xP$	$\forall\ i \leq n, x_i \in Z_q^*,\ Y_i = x_i P$
certificate check	check $Cert$	check $\forall\ i \leq n, Cert_i$
RSU validity check	$\times$	revocation check (by using RL)

To measure the valid number of requesting certificates n_V at once per vehicle, we assume that all OBUs in an RSU range request n_V anonymous certificates to the RSU and the average speed of vehicles varies from 10 m/s $\sim$ 40 m/s (or 36 km/hr $\sim$ 144 km/hr). Therefore, n_V depends on vehicle density d that is computed by 2-second rule which drivers maintain as much distance between vehicles, as the vehicle would travel in 2 s. That is, we have $d = R_{range} \times N_L/(v \times 2)$ where R_{range} is RSU's valid coverage, N_L is the number of lane and v is vehicle speed. Since $T_R \geq T_G \times d$ where $T_R(= R_{range}/v)$ is passing time through an RSU range and $T_G(= 6.3 + 18.6n_V)$ is the time overhead for generating n_V certificates, n_V can be measured as follow:

$$n_V \leq \frac{T_R}{18.6 \times d} - 0.3 = \frac{R_{range}}{v \times 18.6 \times d} - 0.3$$

Therefore, each OBU can request about 10 $\sim$ 50 anonymous certificates to an RSU depending on vehicle density.

Figure 1 shows computational costs of our protocol, ECPP$_M$ and ECPP with different N_R and different n. Then, we can observe that our protocol has reasonable efficiency to ECPP$_M$ in the matter of OBU's computational costs. Furthermore, when compromised RSUs are increasing, both our protocol and ECPP$_M$ which are multiple certificates issue protocols have more efficiency than single certificate issue protocol such as ECPP.

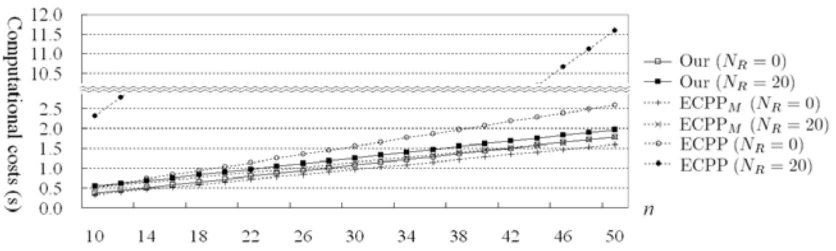


Fig. 1. Computational costs of OBU

Let $|R|_n$ be the minimal number of passed RSUs for n min, and ρ_n be the probability for each OBU to issue a request for n min. Therefore, an OBU could request n certificates to an RSU among $|R|_n$ RSUs. Then, we have $\rho_n = 1/|R|_n$ (note that, since $|R|_n$ is a minimal value, we consider that ρ_n is maximal probability). When we assume that an RSU is allocated every 500 meter on the road, $|R|_1$ and ρ_1 are (10 m/s $\times$ 60 s)/500 m=1.2 and 0.8, respectively. In addition, we have $|R|_n = n \times |R|_1$. As a result, ρ_n can be measured as follow:

$$\rho_n = \frac{1}{|R|_n} = \frac{1}{n \times |R|_1} = \frac{1}{n}\rho_1 = \frac{0.8}{n}$$

By following Lu *et al.*'s analysis method [8], RSU valid serving ratios of the proposed protocol with ECPP$_M$ for $n \geq 10$ are 100% where 10 m/s $\leq v \leq$ 40 m/s, $100 \leq d \leq 400$.

Finally, we show the valid serving ratio of the membership manager for OBUs' requests. In our protocol, the main operation of the membership manager is to decrypt OBUs' pseudonyms at each n min, so the membership manager's performance always depends on the size of n. Then, S_{MM}, the valid serving ratio of the membership manager, can be defined by

$$S_{MM} = \begin{cases} 1, & \text{if } \frac{T_{ms}}{T_{MM} \cdot (T_{avg}/n) \cdot N_{OBU}} \geq 1 \ ; \\ \frac{T_{ms}}{T_{MM} \cdot (T_{avg}/n) \cdot N_{OBU}} & \text{otherwise.} \end{cases}$$

where T_{ms} is a total time for a day, T_{MM} is time overhead for authenticating an OBU, T_{avg} is an average driving time per day and N_{OBU} is the number of OBUs.

From above equation, we can observe that the membership manager can efficiently process multiple certificates requests in most cases where $T_{MM} = 2.1$ ms and $T_{avg} = 120$ min. As a result, the proposed multiple anonymous certificates generation protocol is feasible.

6 Conclusion

In this paper, we have proposed a robust conditional privacy-preserving authentication protocol based on universal re-encryption scheme and identity-based group signature scheme for secure VANET. Compared with ECPP, the proposed protocol can provide unlinkability and traceability even if an attacker compromise multiple RSUs. Furthermore, to avoid frequent certificate requests and to reduce computational overhead for validity check of RSUs in certificate generation phase, RSUs can issue multiple anonymous certificates to an OBU. We have demonstrated, through the performance evaluation, that the proposed protocol has similar performance to ECPP in terms of OBU's computational costs and RSU valid serving ratios.

References

1. Blum, J., Eskandarian, A.: The threat of intelligent collisions. IT Professional, pp 6(1), 22–29 (2004)
2. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: CCS 2004, pp. 168–177 (2004)

3. Freudiger, J., Raya, M., Felegyhazi, M.: Mix-Zones for Location Privacy in Vehicular Networks. In: WiN-ITS 2007 (2007)
4. Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal re-encryption for mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
5. Hubaux, J.-P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Security and Privacy Magazine 2(3), 49–55 (2004)
6. Kamat, P., Baliga, A., Trappe, W.: Secure, pseudonymous, and auditable communication in vehicular ad hoc networks. Security Comm. Networks (1), 233–244 (2008)
7. Luo, J., Hubaux, J.-P.: A survey of Inter-Vehicle Communication Technical Report. EPFL Technical Report IC/2004/24 (2004)
8. Lu, R., Lin, X., Zhu, H., Ho, P.-H., Shen, X.: ECPP: Efficient Conditional Privacy Preservation Protocol for secure Vehicular Communications. In: IEEE INFOCOM 2008, pp. 1903–1911 (2008)
9. Lin, X., Sun, X., Shen, X.: GSIS: a secure and privacy preserving protocol for vehicular communications. IEEE Transaction on Vehicular Technology 56(6), 3442–3456 (2007)
10. Parno, B., Perrig, A.: Challenges in securing vehicular networks. HotNets-IV (2005)
11. Peng, Y., Abichar, Z., Chang, J.M.: Roadside-aided Routing (RAR) in Vehicular Networks. IEEE ICC 2006 8, 3602–3607 (2006)
12. Raya, M., Hubaux, J.-P.: The Security of Vehicular Ad Hoc Networks. In: SASN 2005, pp. 11–21 (2005)
13. Raya, M., Hubaux, J.-P.: Security Aspects of Inter-Vehicle Communications. In: STRC (2005)
14. Raya, M., Hubaux, J.-P.: Securing Vehicle Ad Hoc Networks. Journal of Computer Security 15(1), 39–68 (2007)
15. Ren, K., Lou, W., Deng, R.H., Kim, K.: A Novel Privacy Preserving Authentication and Access Control Scheme in Pervasive Computing Environments. IEEE Transaction on Vehicular Technology 55(4), 1373–1384 (2006)
16. Varsheney, U.: Vehicular mobile commerce. IEEE Computer Magazine Online (2004)
17. Xu, Q., Mak, T., Ko, J., Sengupta, R.: Medium Access Control Protocol Design for Vehicle-Vehicle Safety Messages. IEEE Transaction on Vehicular Technology 56(2), 499–518 (2007)

An Autonomous Attestation Token to Secure Mobile Agents in Disaster Response

Daniel M. Hein and Ronald Toegl

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
{dhein,rtoegl}@iaik.tugraz.at

Abstract. Modern communication and computing devices have the potential to increase the efficiency of disaster response. Mobile agents are a decentralized and flexible technology to leverage this potential. While mobile agent platforms suffer from a greater variety of security risks than the classic client-server approach, Trusted Computing is capable of alleviating these problems. Unfortunately, *Remote Attestation*, a core concept of Trusted Computing, requires a powerful networked entity to perform trust decisions. The existence and availability of such a service in a disaster response scenario cannot be relied upon.

In this paper we introduce the Autonomous Attestation Token (AAT), a hardware token for mobile computing devices that is capable of guaranteeing the trusted state of a limited set of devices without relying on a networked service. We propose a *Local Attestation protocol* with user interaction that in conjunction with the AAT allows to prevent unauthorized access to an emergency mobile agent platform.

Keywords: Disaster Response, Mobile Agents, Trusted Computing, Attestation.

1 Introduction

In recent years, it has been proposed to use agent systems in disaster response [1,2]. Intelligent agents are envisioned to assist and help coordinating teams of robots and humans, thus increasing their efficiency and ultimately helping to save lives. In this scenario agent security is of paramount importance. Unfortunately, due to the nature of mobile agent systems, mobile agent security poses challenges that go far beyond those known from the classic client-server network paradigm.

Mobile agents are self-contained and identifiable computer programs that can move within a network and act on behalf of a user or another entity [3]. Building on the taxonomy of security threats against mobile agents presented by Jansen and Karygiannis [4], we can discern different attacker roles and scenarios. From a high level perspective, those are agent vs. platform, platform vs. agent, agent vs. agent and other vs. agent platform. Specifically, the agent platform should have the characteristics of a reference monitor which employs a set of technologies such as process separation, access control, cryptography, safe code interpretation or proof carrying code [5].

Current mobile agent systems deployed on stationary or mobile devices are not able to proof that they actually employ those security mechanisms and that their configuration has not been tampered with. A promising approach is to apply *Trusted Computing*, as propagated by the *Trusted Computing Group (TCG)*. Balfe and Gallery [6] have shown that Trusted Computing can improve the security of mobile agent systems, by building on security components integrated in off-the-shelf hardware, public key infrastructures, and trusted third parties. However, in a disaster response scenario the latter two services might not be available.

To overcome this challenge, we propose to enhance Trusted Computing based stationary and mobile devices with an additional Autonomous Attestation Token (AAT). This plug-in device additionally provides strong token based authentication for mobile agent systems.

Outline. The remainder of this paper is organized as follows: Outlines of Trusted Computing and mobile agent systems complete Section 1. Section 2 gives an overview of the related work and Section 3 presents the mobile agent systems in a disaster response scenario. Section 4 elaborates the concept of Local Attestation. Section 5 proposes a protocol that prevents access of unauthorized users and compromised systems to an emergency mobile agent system and Section 6 discusses an implementation architecture. The paper concludes in Section 7.

1.1 Trusted Computing

The *Trusted Computing Group*[1] *(TCG)* has specified the *Trusted Platform Module (TPM)* [7] for general purpose computer systems. Similar to a smart card, the TPM features cryptographic primitives, but it is physically bound to its host device. A tamper-resilient integrated circuit contains implementations for public-key cryptography, key generation, cryptographic hashing, and random-number generation.

Likewise, for mobile platforms, the *Mobile Trusted Module (MTM)* [8] has been specified. The MTM standard defines different sets of features (profiles) that implementors can choose from. All of these profiles contain the features required for the purpose outlined in this paper. Therefore, in this paper the terms TPM and MTM are interchangeable.

The TPM implements high-level functionality such as reporting the current system state and providing evidence of the integrity and authenticity of this measurement, known as *Remote Attestation* [9]. This is done with the help of the Platform Configuration Registers (PCRs), which can only be written via the *Extend* operation. A PCR with index i in state t is extended with input x by setting

$$PCR_i^{t+1} = \text{SHA-1}(PCR_i^t || x).$$

Before executable code is invoked, the caller computes the code's hash value and extends a PCR with the result. This process builds a *chain of trust*, starting from the firmware, covering bootloader, kernel, and system libraries etc., up to

[1] http://www.trustedcomputinggroup.org

application code. Ultimately, the exact configuration of the platform is mapped to PCR values. This property makes it impossible to hide a malicious program on a thus protected computer. If such a system state fulfills the given security or policy requirements, we refer to the system state as a *trusted state*.

The TPM is capable of signing the current values of the PCRs together with a supplied nonce. This is called a *Quote* operation. To protect the platform owner's privacy, the unique Endorsement Key, injected by the TPM manufacturer, is not used for this signature. Rather, a pseudonym is used: an *Attestation Identity Key (AIK)*. The authenticity of an AIK can be certified by an online trusted third party, called PrivacyCA [10]. Then a remote *Verifier* can analyze the Quote result and decide whether to trust the given configuration or not.

Another high-level feature of a TPM is that it can *Bind* data (often a symmetric key) to a platform by encrypting it with a *non-migratable* key. Such a key never leaves the TPM's protected storage unencrypted. An extension to this is *Sealing*. A key may be sealed to a specific (trusted) value of the PCRs. A sealed key will not leave the TPM, and it is only used by the TPM, if the PCRs hold the same values that were specified when the key was sealed. Thus, use of the key can be restricted to a single trusted state of the TPM's host computer.

1.2 Mobile Agents

Mobile agents incorporate the concepts of code mobility and user task delegation. Mobile agents are programs that are capable of acting on behalf of a user, and traveling between machines, to better fulfill their tasks. To start the transfer, the mobile agent's code, data, and state is marshaled into a package on the sending machine. This package is then sent to the target machine, where the agent is reconstructed and continues execution. To enable agents to fulfill tasks for their users, they should be autonomous, interactive and adaptable. Autonomy gives them a degree of control over themselves, interactivity allows them to interact with other agents and the environment, and adaptability enables them to react to other agents and the environment. The mobile agents paradigm offers reduction of network traffic and enables asynchronous interaction [11]. A host system, consisting of hardware, firmware, operating system and agent middleware that is capable of executing mobile agents is called agent execution environment. The overall network of agent execution environments together composes a *Mobile Agent Platform (MAP)* in which the agents may roam.

Applications for mobile agents include information gathering, web services, remote software management, network management and many more. Schurr et al. [2] introduce a mobile agent system that helps coordinating fire fights by facilitating vehicle management. This includes planning of routes, resource allocation and even deciding which fire to fight. They combine the agent system with sophisticated visualization technology to enable informed decisions by human personal in a different location. According to Scerri et al. [1] the use of mobile agents to coordinate heterogeneous teams of robots, agents, and humans provides the safest and most effective means for quick disaster response.

2 Related Work

A challenge of Attestation is to report the result of the verification to the user before he unveils confidential information (for example a password) to the system, as malicious software may display fake trust reports. Parno [12] concludes that a local channel between user and TPM is needed.

McCune et al. [13] propose the concept of an *iTurtle* device which can be trusted axiomatically. To achieve *user-observable verification* it should be as simple as possible, and thus easy to understand and certify. McCune et al. propose a USB device with LEDs indicating the trust status. The authors argue that integration of the TCG's scheme was too complex, due to the cryptographic mechanisms involved.

Mobile agent systems suffer from a wider range of security threats than the classic client-server paradigm, aptly described in [4]. These threats are difficult to counter with traditional methods of computer security. Farmer et al. [14] introduce a list of "impossible" security problems for mobile agent systems. The gist is that the MAP has to be trusted to perform as expected.

Besides the TPM, hardware security modules [15] in general have a long history of protecting cryptographic material. However, mobility and integrated support for analysis and verification of a client's state has not been considered yet. Wilhelm et al. [16] propose a specialized, tamper-proof hardware module that provides the so-called trusted processing environment for an agent execution environment. The idea is that if a client trusted the trusted processing environment manufacturer, the trust could automatically be extended to a host with such an environment. Trusted Computing, as proposed by the TCG, has been considered to increase the security of an MAP by [17]. Balfe and Gallery propose specific methods on how Trusted Computing can increase the security of a mobile agent system [6]. SMASH [18] is a trust enhanced MAP that relies on SELinux and Trusted Computing.

3 Mobile Agent Security in Disaster-Relief Scenarios

We assume that after a disaster stroke, one or more response teams are dispatched to the disaster area. The response teams are equipped with mobile computing and communication devices and use a MAP to coordinate their efforts. In this scenario we call these devices *In-Field Devices (IFDs).* Examples for IFDs include general purpose desktop and laptop computers, PDAs and mobile phones. A prerequisite is that IFDs are able to measure their chain-of-trust; therefore being equipped with a TPM. Amongst possible applications for the MAP are the coordination of disaster response work, allocation of resources (like ambulances), or situation visualization.

Response teams may be supported by a command center that, amongst other services, operates a data center and a core network. We assume that this core network is well protected and maintained. In Figure 1 this is referred to as the trusted network. This trusted network may also encompass several trusted servers that execute computational expensive agents.

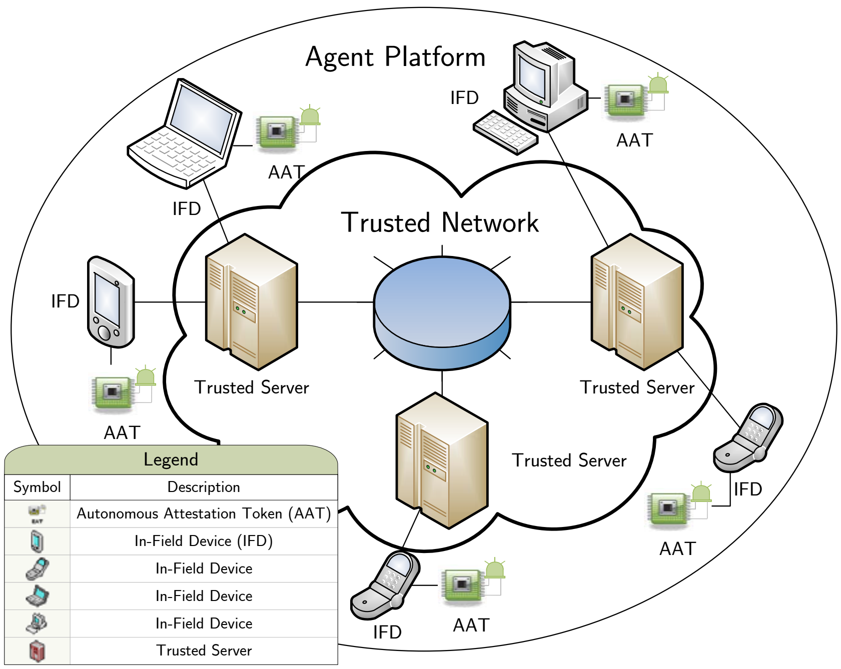


Fig. 1. Mobile agent platform for disaster response scenario

We assume that the IFDs are capable of connecting to an Internet Protocol based network. Establishing connectivity in emergency situations, possibly over multiple bearers, is by no means an easy task. Therefore, we must assume that at times the connection to the core network will be interrupted. Even worse, a core network might not exists at all, because the command center is not yet established. In this case, IFDs will form local or ad-hoc networks to support key disaster response personal.

Regardless of the underlying network topology, we assume that the MAP uses a secure network that connects the different agent execution environments. Furthermore, we assume that access to this network is secured by a single cryptographic key.

Security in such a scenario is of paramount importance. A malicious entity could cause untold harm by attacks. Attacks could encompass the intentional redirection of resources like ambulance vehicles or teams of fire fighters to wrong locations. The danger of malicious entities is especially prominent if the disaster was caused by deliberate actions, like arson or a terrorist attack.

The access to the MAP must therefore be solely restricted to authenticated and authorized personnel. In addition to a basic authentication mechanism we identify the need for Trusted Computing mechanisms to protect against tampered IFDs. It is highly desirable to enable the protection of a platform against software attacks, thus ensuring its configuration and behavioral integrity. With

TPM equipped systems, it is possible to build a chain of trust. This chain of trust does not prevent execution of malicious software, but makes all malicious software on a system evident in the system Quote.

However, current TCG-based implementations of the Remote Attestation mechanism rely on online, *remote* servers to certify the identity of the IFD, verify the authenticity of system reports, analyze the configuration of the IFD, and display the security status to the user.

The requirement for a remote party is in direct conflict with the challenges faced in disaster response, as connections to servers may fail and also no second device may be available to display the IFD's trust state.

Sealing, the TCG's alternative mechanism, may also be used off-line, but it is too inflexible: data or agent code can only be sealed to a single configuration of a single platform. When configurations change, for example by migrating agents, a central authority would be needed to assess the trusted state of the new platform and migrate the data.

For mobile platforms only, the TCG describes *Secure Boot* [19]. Secure Boot builds not only a chain-of-trust, but it also verifies intermediate states and aborts the boot process, if the system has been compromised. However, Secure Boot is restricted to a single configuration of a single device and the process is not under direct control of the user.

Usability, next to security, is an important requirement, especially in a disaster response scenario. It cannot be expected of disaster relief personal to remember twenty character passwords or to employ similar complex security measures. Therefore, the solution must provide high security with only moderate demands on usability. Technically, high security amounts to a cryptographic MAP access key with sufficient length (for example a 256-bit AES key). We believe that plugging in a security token and entering a short four to six character password is acceptable and can also be performed in crisis situations.

4 Local Attestation

As outlined above, standard Trusted Computing mechanisms building on the TPM alone cannot fulfill the security requirements in disaster response scenarios. We propose to extend these concepts by introducing a second hardware security token, the *Autonomous Attestation Token (AAT).* We envision the AAT to be either the size and format of a smart card or a microSD card. This allows to plug it into any available device, be it a laptop computer or a mobile phone.

A hardware security token like the AAT could potentially provide various cryptographic services. In the following sections we describe a basic use case where the AAT protects the access key to an emergency MAP in a specially shielded hardware storage unit. The key is released to the requester, if and only if the requester can provide sufficient authentication and evidence that it is in a trusted state. As the user needs to be physically present, we use the term *Local Attestation* to designate the process of deciding if a requester, in this case the IFD, is in a trusted state using our proposed AAT as verifier.

The mobile attestation token implements a true, decentralized trust decision process, which does not rely on third parties. It can easily accommodate different platforms with varying valid configurations, and it can implement a direct feedback channel to the user. The AAT, in addition to performing the platform state verification, also serves as a proof of possession in authentication protocols. Therefore, it not only ensures a specific state of the platform, but also provides evidence of the identity of the owner.

4.1 Operational Challenges

The Local Attestation approach poses several challenges related to the deployment and maintenance of AATs. Deployment is concerned with issuing an AAT to every entity that requires access to the MAP. Every AAT must contain a set of valid configurations for all devices of its owner. In addition to this, it must also contain the public part of the Attestation Identity Keys of said devices. These keys are used to sign the system Quotes of these devices.

Deployment includes revocation of an AAT. For our proposed scenario (cf. Section 3), we presume that there is no permanent central authority that grants access to the emergency MAP, thus establishing the need for Local Attestation. Local Attestation enables IFDs to ensure continuing collaboration on a distributed, yet trusted MAP. In such a case, there is also no central authority that can revoke access to the MAP. Therefore, we must rely on operational procedures, that is physically taking away the AAT from its owner.

The maintenance challenge stems from the nature of platform configuration measurements as defined by the TCG. A measurement is a SHA-1 hash of an application that is computed before the application is executed. Therefore, every time a software component of an IFD is changed, for example during an update, its hash value changes. This change must be reflected by the known-good-values database of the verifier.

We believe that careful operational practices, as they can be expected of emergency response organizations, will allow to overcome these challenges in practice.

5 An Attestation Based Key Release Protocol

In the emergency MAP scenario given above the AAT protects a single cryptographic key or a set of keys that grant access to this MAP. The objective of the protocol introduced in this work is to protect the MAP against misuse by unauthorized or malicious entities or maliciously modified platforms of authorized users. In order to achieve this goal the protocol relies on both Local Attestation and immediate user interaction. An interesting feature of our proposed protocol is that it employs a token specific return channel in the shape of a red and a green LED, similar to iTurtle proposed by McCune et al. [13].

On request by its user, the IFD initiates the connection procedure to the MAP emergency network. To achieve this end, it requires the protected key material from the AAT. The detailed key release protocol flow is illustrated in Figure 2.

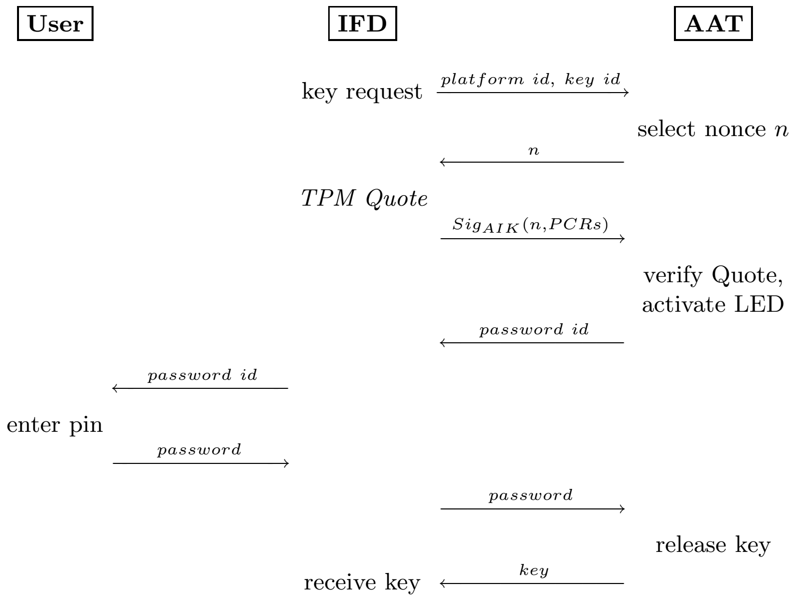


Fig. 2. Key release protocol based on Local Attestation and user interaction

Initially, the IFD sends a key request containing the unique identifier (*key id*) of the required key and a platform identifier (*platform id*). The platform identifier uniquely identifies the platform type, thus facilitating the verification process. In response to this the AAT answers with a nonce n.

The IFD forwards n to the TPM, which incorporates it in the generation of a Quote of the platform configuration. The Quote is signed by the TPM using the private part of an Attestation Identity Key (AIK). The AIK must be premeditated and the corresponding public key must be stored on the AAT. When the Quote is analyzed by the AAT, the signature, the nonce and the platform configuration report are checked. If both the nonce and the signature are acceptable, the actual trust decision is made based on the Platform Configuration Register values reflected in the Quote. This is simply a comparison of the values with a known-good-values database.

If the AAT finds the IFD trustworthy, it sends a password identification (*password id*). At the same time the AAT turns on a green LED that signals to the user that the platform is in a trusted state. In the case that the IFD is not in a trusted state, the AAT signals this by flashing a red LED. The LEDs guarantee a direct, trustworthy, one-way channel to the user.

After the trust in the IFD has been established and the password identification is sent to it, it prompts the user to enter the password for the given identification. The user checks the LED status. Only if the green LED glows, she entrusts the IFD with the required password. This password is then forwarded to the AAT which finally releases the connection key to the IFD, if the password is correct.

This protocol is immune to the general Cuckoo attack. This attack is also called grandmaster postal chess problem, and describes a technique where a malicious party forwards requests to any entity that is capable of answering them correctly [20]. A universal solution for this problem is an open research topic [12,13,21]. By requiring premeditation of the AIKs between the IFD and the AAT, it is not possible to forward the Attestation part of this protocol to just any other IFD. The Cuckoo attack can only work if this part is forwarded to one of the valid devices of the same user. Therefore, for a successful attack it is necessary to physically steal one of the emergency communication devices of the AAT owner. Furthermore, the user must be given a modified device that forwards the Attestation part of the protocol. Also it is not sufficient to steal just an AAT because of the specific AIKs and the need of a user supplied secret. Therefore, our proposed protocol increases the security system by providing protection against software attacks on the IFD.

6 The AAT Hardware Architecture

In this section, we propose a hardware architecture that is capable of implementing the Local Attestation protocol. The emergency MAP application scenario requires the AAT to cooperate with mobile devices. This influences the form factor and interface of the AAT (microSD/smart card), as well as the digital hardware design properties such as die area and circuit speed. The AAT is only necessary to establish the initial connection to the MAP, that is once per power cycle. Therefore, we conclude that circuit speed and power consumption are insignificant factors in this scenario. Verification cycles of two or three seconds seem acceptable and the only requirement on power consumption is that it has to stay below the limit of what a mobile host device can supply. This can be utilized to reduce the die area, an important factor in the total cost of the circuit. This simplifies AAT design efforts, as they can be concentrated on tamper resistance, and area minimization. The high level hardware architecture of the AAT we propose is depicted in Figure 3.

The AAT hardware architecture is composed of an RSA signature engine, a SHA-1 hash engine, a true random number generator, a Static Random Access Memory (SRAM), a Non-Volatile RAM (NV-RAM), a processor, a red LED, and a green LED. The RSA signature engine and the SHA-1 hash engine are required for the verification of the signature on the TPM Quote. The RSA engine must accommodate the same key lengths as a TPM (typically 2048 bits). The random number generator, which uses a physical randomness source with bias correction is necessary for the generation of fresh nonces. The NV-RAM contains both the secret key(s) that must be protected, and the known-good-values required for verification of the host platform state. Therefore, the AAT in general and the NV-RAM specifically must be shielded against tamper attacks [15]. The tamper resilience must be equal or better than the tamper resilience of the host platforms TPM, otherwise the AAT would provide a point of attack.

The actual verification of the host platform configuration is executed by the processor in conjunction with the SRAM. The processor also serves as controlling

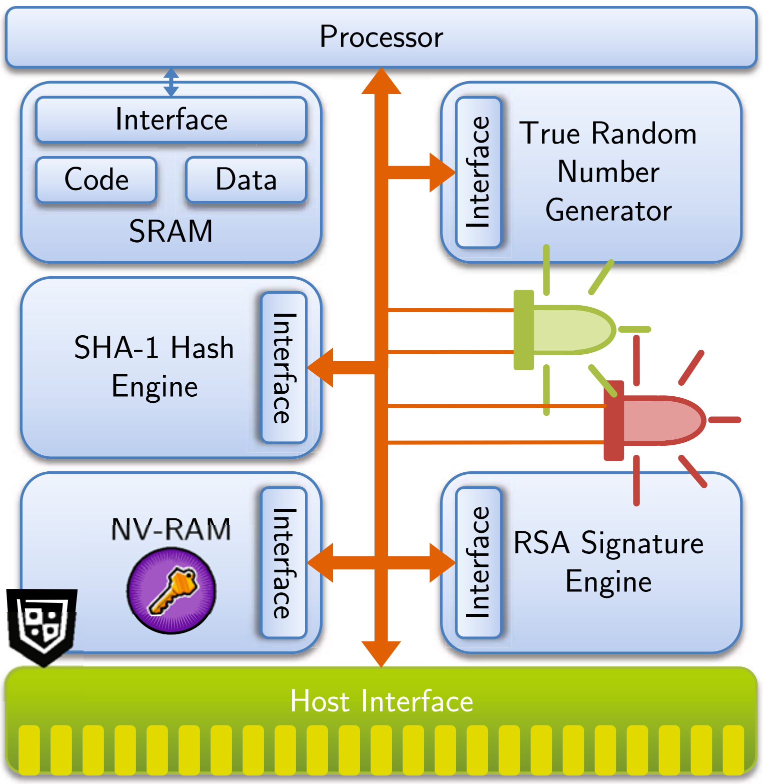


Fig. 3. The AAT architecture

instance for the complete device. For an Attestation based on known-good-values a simple 8-bit microcontroller is sufficient. The protocol supplies a platform type identifier which confines the relevant known-good-values set, thus simplifying the verification.

The architecture introduced above represents a minimal set of components to implement an AAT. Actually, the AAT functionality can be implemented on any programmable smart card or hardware security module that provides the necessary components and possesses an interface compatible with mobile devices.

7 Conclusion

Mobile agent systems and disaster response are a mixture that combines high security requirements with security risks that exceed those of the client-server paradigm. Despite this fact, the advantages and applications of this combination render it desirable. Current research indicates that Trusted Computing might provide solutions to the various security problems of mobile agent systems. In this paper we propagate the use of a Local Attestation token, the AAT, to eliminate the need for a Trusted Computing specific remote verifier and allow decentralized, possibly ad-hoc mobile agent cooperation. Expanding on the idea of the AAT, we introduce a key release protocol that unites Attestation, a separate feedback channel, and user interaction to limit agent platform access to authorized users only. The protocol provides security against remote or local software attacks on an in-field device and is immune to a general Cuckoo attack. Furthermore, we argue that a hardware implementation of the AAT is relatively simple compared to full-featured hardware security modules or even some smart cards and propose a possible architecture. Thus, our AAT provides a powerful and simple to use security solution for mobile agents in disaster response scenarios.

Acknowledgments. The work reported in this paper was supported by the European Commision through projects SECRICOM, FP-7, contract no. FP7-SEC-218123 and OpenTC, FP-6, contract no. 027635.

References

1. Scerri, P., Pynadath, D., Johnson, L., Rosenbloom, P., Si, M., Schurr, N., Tambe, M.: A prototype infrastructure for distributed robot-agent-person teams. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 433–440. ACM, New York (2003)
2. Schurr, N., Marecki, J., Tambe, M.: The future of disaster response: Humans working with multiagent teams using DEFACTO. In: AAAI Spring Symposium on AI Technologies for Homeland Security (2005)
3. Rothermel, K., Schwehm, M.: Mobile agents. In: Proceedings of the 1st International Workshop, pp. 155–176. Springer, Heidelberg (1997)
4. Jansen, W., Karygiannis, T.: NIST special publication 800-19 - mobile agent security (2000)
5. Necula, G.C., Lee, P.: Untrusted agents using proof-carrying code. In: Vigna, G. (ed.) Mobile Agents and Security. LNCS, vol. 1419, pp. 61–91. Springer, Heidelberg (1998)
6. Balfe, S., Gallery, E.: Mobile agents and the deus ex machina. In: 21st International Conference Advanced Information Networking and Applications Workshops (AINAW 2007), May 2007, vol. 2, pp. 486–492 (2007)
7. Trusted Computing Group: TCG TPM specification version 1.2 revision 103 (2007), https://www.trustedcomputinggroup.org/specs/TPM/
8. Trusted Computing Group: TCG mobile trusted module specification version 1.0 revision 6 (June 2008), https://www.trustedcomputinggroup.org/specs/mobilephone/
9. Coker, G., Guttman, J.D., Loscocco, P., Sheehy, J., Sniffen, B.T.: Attestation: Evidence and trust. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 1–18. Springer, Heidelberg (2008)
10. Pirker, M., Toegl, R., Hein, D., Danner, P.: A PrivacyCA for anonymity and trust. In: TRUST 2009. LNCS, vol. 5471. Springer, Heidelberg (2009)
11. Pham, V.A., Karmouch, A.: Mobile software agents: an overview. IEEE Communications Magazine 36(7), 26–37 (1998)
12. Parno, B.: Bootstrapping trust in a "trusted" platform. In: HOTSEC 2008: Proceedings of the 3rd conference on Hot topics in security, Berkeley, CA, USA, USENIX Association, pp. 1–6 (2008)

13. McCune, J.M., Perrig, A., Seshadri, A., van Doorn, L.: Turtles all the way down: Research challenges in user-based attestation. In: Proceedings of the Workshop on Hot Topics in Security (HotSec) (August 2007)
14. Farmer, W., Guttman, J., Swarup, V.: Security for mobile agents: Issues and requirements. In: National Information Systems Security Conference, NISSC 1996 (1996)
15. Anderson, R., Bond, M., Clulow, J., Skorobogatov, S.: Cryptographic processors-a survey. Proceedings of the IEEE 94(2), 357–369 (2006)
16. Wilhelm, U., Staamann, S., Buttyan, L.: Introducing trusted third parties to the mobile agent paradigm. In: Vitek, J., Jensen, C. (eds.) Secure Internet Programming. LNCS, vol. 1603, pp. 471–491. Springer, Heidelberg (1999)
17. Wu, X., Shen, Z., Zhang, H.: The mobile agent security enhanced by trusted computing technology. In: International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2006), September 2006, pp. 1–4 (2006)
18. Pridgen, A., Julien, C.: SMASH: Modular security for mobile agents. In: Choren, R., Garcia, A., Giese, H., Leung, H.-f., Lucena, C., Romanovsky, A. (eds.) SELMAS. LNCS, vol. 4408, pp. 99–116. Springer, Heidelberg (2007)
19. Dietrich, K., Winter, J.: Secure boot revisited. In: TrustCom 2008 Proceedings, in ICYCS Proceedings, pp. 2360–2365 (2008)
20. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (2001)
21. Stumpf, F., Tafreschi, O., Röder, P., Eckert, C.: A robust integrity reporting protocol for remote attestation. In: Proceedings of the Second Workshop on Advances in Trusted Computing, WATC 2006 (Fall 2006)

An ECDLP-Based Threshold Proxy Signature Scheme Using Self-Certified Public Key System*

Qingshui Xue[1], Fengying Li[1, 2], Yuan Zhou[3], Jiping Zhang[3], Zhenfu Cao[4], and Haifeng Qian[5]

[1] School of Techniques, Shanghai Jiao Tong University, 201101, Shanghai, China
xue-qsh@sjtu.edu.cn
[2] Dept. of Education Information Technology, East China Normal University, 200062, Shanghai, China
fyli@sjtu.edu.cn, jpzhang@deit.ecnu.edu.cn
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China
zhouyuantdt@163.com
[4] Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, 200240, Shanghai, China
zfcao@cs.sjtu.edu.cn
[5] Dept. of Computer Science and Technology, East China Normal University, 200062, Shanghai, China
hfqian@cs.ecnu.edu.cn

Abstract. In a (t, n) threshold proxy signature scheme, one original signer delegates a group of n proxy signers to sign messages on behalf of the original signer. When the proxy signature is created, at least t proxy signers cooperate to generate valid proxy signatures and any less than t proxy signers can't cooperatively generate valid proxy signatures. So far, all of proposed threshold proxy signature schemes are based on public key systems with certificates, which have some disadvantages such as checking the certificate list when needing certificates. Most threshold proxy signature schemes use Shamir's threshold secret share scheme. Identity-based public key system is not pretty mature. Self-certified public key systems have attracted more and more attention because of its advantages. Based on Hsu et al's self-certified public key system and Li et al's proxy signature scheme, one threshold proxy signature scheme based on ECDLP and self-certified public key system is proposed. As far as we know, it is the first scheme based on ECDLP and self-certified public key system. The proposed scheme can provide the security properties of proxy protection, verifiability, strong identifiability, strong unforgeability, strong repudiability, distinguishability, known signers and prevention of misuse of proxy signing power. That is, internal attacks, external attacks, collusion attacks, equation attacks and public key substitution attacks can be resisted. In the proxy signature verification phase, the authentication of the original and the proxy signers' public keys and the verification of the threshold proxy signature are executed together. In addition, the computation overhead and communication cost of the proposed scheme are analyzed as well.

* This paper is supported by the National Natural Science Foundation of China under Grant No. 60673079 and 60873217, and National Basic Research Program of China (973 Program) under Grant No.2007CB311100.

1 Introduction

The proxy signature scheme [1], a variation of ordinary digital signature schemes, enables a proxy signer to sign messages on behalf of the original signer. Proxy signature schemes are very useful in many applications such as electronics transaction and mobile agent environment.

Mambo et al. [1] provided three levels of delegation in proxy signature: full delegation, partial delegation and delegation by warrant. In full delegation, the original signer gives its private key to the proxy signer. In partial delegation, the original signer produces a proxy signature key from its private key and gives it to the proxy signer. The proxy signer uses the proxy key to sign. As far as delegation by warrant is concerned, warrant is a certificate composed of a message part and a public signature key. The proxy signer gets the warrant from the original signer and uses the corresponding private key to sign. Since the conception of the proxy signature was brought forward, a lot of proxy signature schemes have been proposed [2]-[19].

Recently, many threshold proxy signature schemes were proposed [2] [6]-[14]. In threshold proxy signature schemes, a group of n proxy signers share the secret proxy signature key. To produce a valid proxy signature on the message m, individual proxy signers produce their partial signatures on that message, and then combine them into a full proxy signature on m. In a (t,n) threshold proxy signature scheme, the original signer authorizes a proxy group with n proxy members. Only the cooperation of t or more proxy members is allowed to generate proxy signatures. Threshold signatures are motivated both by the demand which arises in some organizations to have a group of employees agree on a given message or document before signing, and by the need to protect signature keys from attacks of internal and external adversaries.

In 1999, Sun proposed a threshold proxy signature scheme with known signers [9]. Then Hwang et al. [7] pointed out that Sun's scheme was insecure against collusion attack. By the collusion, any $t-1$ proxy signers among t proxy signers can cooperatively obtain the secret key of the remainder one. They also proposed an improved scheme which can guard against the collusion attack. After that, [6] showed that Sun's scheme was also insecure against the conspiracy attack. In the conspiracy attack, t malicious proxy signers can impersonate some other proxy signers to generate valid proxy signatures. To resist the attack, they also proposed a scheme. Hwang et al pointed out [8] that the scheme in [7] was also insecure against the attack by the cooperation of one malicious proxy signer and the original signer. In 2002, Li et al. [2] proposed a threshold proxy signature scheme with good properties and performance.

All of the proposed schemes are based on Shamir's secret share protocol and the public key systems using certificates, which have some disadvantages such as checking the certificate list when needing certificates, and high computation overheads and communication cost.

So far, there are three kinds of public key systems involving using certificates, identity-based and self-certified public keys. Currently, identity-based public systems are not mature, as makes it not used in the real life.

The self-certified public key system was first introduced by Girault in 1991 [20]. In self-certified public key systems, each user's public key is produced by the **CA** (Certification Authority), while the corresponding private key is only known to the user. The authenticity of public keys is implicitly verified without the certificate. That is, the verification of public keys can be performed with the subsequent cryptographic applications such as key exchange protocols and signature schemes in a single step. Compared with other two public systems, the system has the following advantages [18]: ①the storage space and the communication overheads can be reduced since the certificate is not needed; ②the computation overhead can be reduced as it doesn't require public key verification.

In public key cryptosystems, there are several kinds of cryptographic assumptions to be used. Currently, only the discrete logarithm problem and factorization problem are widely accepted. In addition, the elliptic curve cryptosystem (ECC) [21] is constructed by integer points over elliptic curves in finite fields. The advantage of ECC is that it can reach the same level of security constituted by DSA [22] or RSA [23] and provides better efficiency than both discrete logarithm and factorization systems.

There are four trust levels for the security of public key systems [12]. Hsu et al [18] pointed out that the self-certified public key systems might be the ideal choice for realizing cryptographic applications according to security and efficiency. Further, Hsu et al [18] proposed a kind of self-certified public key system. In 2004, Hwang et al [19] proposed a generation of proxy signature based on elliptic curves. In the paper, based on Hsu et al's self-certified public key system and Hwang's et al's proxy signature scheme, a threshold proxy signature scheme using self-certified public system is proposed by us. The main advantage of the proposed scheme is that the authenticity of the original and the proxy signers' public keys, and the verification of the proxy signature can be simultaneously executed in a single step. As far as we know, this threshold proxy signature scheme is the first one using self-certified public key system.

In the paper, we will organize the content as follows. In section 2, we will detail the proposed threshold proxy signature scheme, which is based on the self-certified public key system [18] and Hwang et al's proxy signature scheme [17]. The security of the proposed scheme will be analyzed and discussed in section 3. In section 4, we will analyze the computational overheads and communication cost of the proposed scheme. Finally, the conclusion is given.

2 The Proposed Scheme

In the scheme, a system authority (**SA**) whose tasks are to initialize the system, the original signer U_O, certification authority (**CA**) whose tasks are to generate the public key for each user, the proxy group of n proxy signers $G_P = \{U_{P_1}, U_{P_2}, ..., U_{P_n}\}$, one designated clerk **C** whose tasks are to collect and verify the individual proxy signatures generated by the proxy signers, and construct the final threshold proxy signature, and the signature verifier are needed.

Throughout the paper, q is a large prime and E is an elliptic curve over a finite $GF(q)$. G is a base point on E with order n. h is a secure one-way hash function. The parameters (q,G,n) and the function h are made public. Let ID_i be the identifier of the user U_i. Assume that x_{CA} and y_{CA} are the private and public keys of the **CA**, respectively, where $x_{CA} \in Z_q^*$ and

$$y_{CA} = x_{CA}G \qquad (1)$$

m_w is a warrant which records the identities of the original signer and the proxy signers of the proxy group, parameters t and n, message type to sign, the valid delegation time, etc. *ASID* (Actual Signers' *ID*) denotes the identities of the actual proxy signers.

The proposed scheme consists of four phases: registration, proxy share generation, proxy signature issuing without revealing proxy shares and proxy signature verification. We will detail them as follows.

2.1 Registration

Step 1. Each user U_i selects an integer $1 \le t_i \le n-1$ at random, computes

$$v_i = h(t_i, ID_i)G \qquad (2)$$

and sends (v_i, ID_i) to the **CA**.

Step 2. Upon receiving (v_i, ID_i) from U_i, the **CA** selects $1 \le z_i \le n-1$, calculates

$$y_i = v_i + z_iG \qquad (3)$$

$$e_i = z_i + h((y_i)_x, ID_i)x_{CA} \bmod n \qquad (4)$$

and returns (y_i, e_i) to U_i. Here $(\cdot)_x$ denotes the x-coordinate of point $(\cdot)$ on E.

Step 3. U_i computes

$$x_i = e_i + h(t_i, ID_i) \bmod n \qquad (5)$$

and confirms its validity by checking that

$$y_{CA}h((y_i)_x, ID_i) + y_i = x_iG \qquad (6)$$

If it holds, U_i accepts (x_i, y_i) as his private and public keys. Moreover, the **CA** publishes U_i's public key y_i when the registration is complete. Note that the **CA** needn't issue extra certificate associated with y_i.

2.2 Proxy Share Generation

Step 1. The original signer chooses randomly an integer $1 \le k_o \le n$, computes

$$K_o = k_oG \qquad (7)$$

$$\sigma_o = k_o(K_o)_x + x_o h(m_w,(K_o)_x) \bmod n \tag{8}$$

and sends (m_w, K_o, σ_o) to each of proxy signers.

Step 2. After receiving (m_w, K_o, σ_o), each of proxy signers confirms the validity of (m_w, K_o, σ_o) by

$$\sigma_o G = (K_o)_x K_o + h(m_w,(K_o)_x)[y_{CA} h((y_o)_x, ID_o) + y_o] \tag{9}$$

If it holds, each of proxy signers regards σ_o as its proxy share.

2.3 Proxy Signature Issuing without Revealing Proxy Shares

Without loss of generality, the proposed scheme allows any *t* or more proxy signers to represent the proxy group to sign a message *m* cooperatively on behalf of the original signer U_o.

Let $G_{P'} = \{U_{P_1}, U_{P_2}, ..., U_{Pt'}\}$ be the actual proxy signers for $t \le t' \le n$. $G_{P'}$ as a group performs the following steps to generate a threshold proxy signature.

Step 1. Each proxy signer $U_{P_i} \in G_{P'}$ chooses an integer $k_i \in Z_q^*$ at random, computes

$$K_i = k_i G \tag{10}$$

and sends it to the other $t'-1$ proxy signers in $G_{P'}$ and the designated clerk **C**.

Step 2. Upon receiving K_j $(j = 1,2,...,t'; j \ne i)$, each $U_{P_i} \in G_{P'}$ computes *K* and s_i as follows:

$$K = \sum_{j=1}^{t'} K_j \tag{11}$$

$$s_i = k_i(K)_x + (\sigma_o t'^{-1} + x_{P_i}) h(m, ASID) (\bmod n) \tag{12}$$

Here, s_i is an individual proxy signature which is sent to **C**.

Step 3. For each received s_i $(i = 1,2,...,t')$, **C** checks whether the following congruence holds:

$$\begin{aligned} s_i G = K_i(K)_x + \{t'^{-1}[(K_o)_x K_o + h(m_w,(K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + \\ [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\} h(m, ASID) \end{aligned} \tag{13}$$

If it holds, (K_i, s_i) is a valid individual proxy signature on *m*. If all the individual proxy signatures of *m* are valid, the clerk **C** computes

$$S = \sum_{i=1}^{t'} s_i \bmod n \tag{14}$$

Then, $(m_w, K_o, m, K, S, ASID)$ is the proxy signature on *m*.

2.4 Proxy Signature Verification

After receiving the proxy signature $(m_w, K_o, m, K, S, ASID)$ for *m*, any verifier can verify the validity of the threshold proxy signature by the following steps.

Step 1. According to m_w and $ASID$, the verifier can obtain the value of t and n, the public keys of the original signer and proxy signers from **CA** and knows the number t' of the actual proxy signers. Then the verifier checks whether $t' \geq t$, if it holds, he/she continues the following steps, or else, he/she will regard the threshold proxy signature $(m_w, K_o, m, K, S, ASID)$ invalid.

Step 2. The verifier confirms the validity of the proxy signature on m by checking

$$SG = K(K)_x + h(m, ASID)\{[(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + \sum_{i=1}^{t'} [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\} \tag{15}$$

If it holds, the proxy signature $(m_w, K_o, m, K, S, ASID)$ is valid.

3 Correctness of the Proposed Scheme

In the section, we shall prove that the proposed scheme can work correctly by the following theorems.

Theorem 1. *In the registration phase, the user U_i can verify the validity of its private and public key pair (x_i, y_i) by Equation (6).*

Proof. From Equations (4) and (5), we have $x_i = z_i + h((y_i)_x, ID_i)x_{CA} + h(t_i, ID_i) \bmod n$. By raising both sides of the above equation by multiplying them by the base point G, we have

$$x_i G = z_i G + h((y_i)_x, ID_i)x_{CA} G + h(t_i, ID_i)G .$$

From Equations (2) and (3), we have $z_i G = y_i - h(t_i, ID_i)G$. Then, we have

$$x_i G = y_i - h(t_i, ID_i)G + h((y_i)_x, ID_i)x_{CA} G + h(t_i, ID_i)G = y_i + h((y_i)_x, ID_i)x_{CA} G$$

From Equation (1), the above equation can be rewritten as

$$y_{CA} h((y_i)_x, ID_i) + y_i = x_i G$$

Theorem 2. *If the proxy share is constructed correctly, it will pass the verification of Equation (9).*

Proof. By raising both sides of Equation (8) by multiplying them by the base point G, we have

$$\sigma_o G = k_o (K_o)_x G + x_o h(m_w, (K_o)_x)G$$

According to Equations (6) and (7), the above equation can be rewritten as

$$\sigma_o G = (K_o)_x K_o + h(m_w, (K_o)_x)[y_{CA} h((y_o)_x, ID_o) + y_o]$$

Theorem 3. *In the proxy signature generation phase, the clerk* **C** *can verify any individual proxy signature s_i sent from U_{P_i} by Equation (13).*

Proof. By raising both sides of Equation (12) by multiplying them by the base point G, we have

$$s_i G = k_i (K)_x G + (\sigma_o t'^{-1} G + x_{P_i} G) h(m, ASID)$$

According to Equations (6), (9) and (10), the above equation can be rewritten as

$$s_i G = K_i (K)_x + \{t'^{-1}[(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\} h(m, ASID)$$

Theorem 4. *If the threshold proxy signature is constructed correctly, it will pass the verification of Equation (15).*

Proof. From Equations (12) and (14), we have

$$S = \sum_{i=1}^{t'} [k_i (K)_x + (\sigma_o t'^{-1} + x_{P_i}) h(m, ASID)] (\mathrm{mod}\, n)$$
$$= \sum_{i=1}^{t'} k_i (K)_x + \sigma_o h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} h(m, ASID) (\mathrm{mod}\, n)$$

By raising both sides of the above equation by multiplying them by the base point G, we have

$$SG = \sum_{i=1}^{t'} k_i G (K)_x + \sigma_o G h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} G h(m, ASID)$$

According to Equations (6), (9), (10) and (11), the above equation can be rewritten as

$$SG = K(K)_x + h(m, ASID)\{[(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + \sum_{i=1}^{t'} [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\}$$

4 Security Analysis

In the section, we will propose several theorems about the security below and prove that they are right.

Theorem 5. *The user can't forge his/her private key without interaction with the* **CA** *and the* **CA** *can forge the user's public key without the interaction with the user neither.*

Proof. From Equation (4), we know that although the user can select a random integer $z_i \in Z_q^*$ and compute $y_i = v_i + z_i G$, because having no the knowledge of the **CA**'s private key x_{CA}, he/she can't get a valid value of e_i to construct his self-certified private key. Obviously, the user can forge a valid private key with the probability of $1/n$. That's, the user's private key has to be set up by the interaction with the **CA**.

Similarly, if the **CA** wants to forge the user's new public key which satisfies Equation (6), he/she has to solve the difficult discrete logarithm problem and the secure hash function, as we know it is impossible. Thus the **CA** can't forge a new public key of the user. To generate the user's public key, the **CA** has to interact with the user.

Theorem 6. *The user can't forge his public key by its private key without the interaction with the* **CA** *and the* **CA** *can't get the user's private key from the interaction with the user either.*

Proof. If the user wants to forge his/her new public key which satisfies Equation (6), he/she has to solve the difficult discrete logarithm problem and secure hash functions, as we know it is impossible. Thus the user can't forge a new public key of the user without the interaction with the **CA**. From Equation (5), we know that because the **CA** has no the knowledge of $t_i \in Z_q^*$ selected by the user, the **CA** can't obtain the user's private key x_i. In addition, from the verification equation (6), the **CA** is unable to get the user's private key x_i since he/she is faced with the difficulty of solving discrete logarithms and secure hash functions. Therefore, we can draw the above conclusion.

Theorem 7. *Any $t'' < t$ proxy signers can't generate a valid threshold proxy signature on a new message m'.*

Proof. From Equations (12) and (14), we have

$$S = \sum_{i=1}^{t'} k_i (K)_x + \sigma_o h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} h(m, ASID) (\bmod n) \tag{16}$$

Because any $t'' < t$ proxy signers have no the knowledge of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers, any $t'' < t$ proxy signers are unable to cooperate to generate the valid proxy signature on a new message m'. Although any $t'' < t$ proxy signers can generate $(m_w, K_o, m', K, S, ASID)$ and it can pass the verification Equation (15), the number of actual proxy signers is less than *t*, as makes it can't pass the verification step 1. Thus the forged proxy signature $(m_w, K_o, m', K, S, ASID)$ by $t'' < t$ proxy signers is invalid.

From the Equation (16), any $t'' < t$ proxy signers are unable to get the values of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers, if the message m is replaced with m', any $t'' < t$ proxy signers can't get the new value of S'. Also, from the verification Equation (15), when m is replaced with m', given fixed some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will be unable to be gotten because of the difficult discrete logarithm and secure hash function. That is, from a known proxy signature $(m_w, K_o, m, K, S, ASID)$, any $t'' < t$ proxy signers can't generate valid threshold proxy signature on a new message m'. So the theorem is proved to be true.

Theorem 8. *Any $t'' < t$ proxy signers can't forge another valid threshold proxy signature on the original message m from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. On one hand, from Equation (16), any $t'' < t$ proxy signers can't get the knowledge of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers. Thus, the values of some variables in set $\{K, S, ASID\}$ can't be changed by changing the values of the other variables in set $\{K, S, ASID\}$. Here note that as far as any $t'' < t$ proxy signers are concerned, the values of m_w and K_O can't be changed, as can be guaranteed by Equation (9). On the other hand, from the verification Equation (15), by fixing some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will not be able to be gotten because of the difficult discrete logarithm and secure hash functions. Therefore, the theorem is proved true.

Theorem 9. *The original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message* m'.

Proof. The case is similar with Theorem 7. The difference is that the original signer is one of the forgers. First, if the original signer does not change the values of m_w and K_O, the difficulty of forging a valid proxy signature on m' is equivalent to that of Theorem 7 since the original signer also has no the knowledge of x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers. Second, if the original signer changes the values of m_w and K_O, correspondingly, the proxy share σ_o is also changed, however, the values of t and n should not be changed, as is obvious in the case. Thus the condition is similar with that of the first case. Therefore, the original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message m'.

Theorem 10. *The original signer and any $t'' < t$ proxy signers can't cooperatively forge another valid threshold proxy signature on the original message m from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. The theorem is similar with Theorem 8. The first condition is similar with that of the proof of Theorem 8. Here the kind of proof is omitted. Let us see the second condition. From the verification Equation (15), by fixing some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will not be able to be obtained because of the difficult discrete logarithm and secure hash function, although the original signer is easy to change the values of m_w and K_O. Thus the theorem holds.

Theorem 11. *Any third party, the original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message* m'.

Proof. The theorem is the same as Theorem 9 since the third party knows less information than the original signer. The proof is the same as that of the Theorem 9.

Theorem 12. *Any third party, the original signer and any $t'' < t$ proxy signers can't cooperatively forge another valid threshold proxy signature on the original message m from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. The theorem is the same as Theorem 10 since the third party knows less information than the original signer. The proof is the same as that of the Theorem 10.

Theorem 13. *Any can be convinced of the original signer's agreement on the signed message from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. From the proxy signature verification equation (15), the warrant m_w, the identities and public keys of the original and actual proxy signers are used. In this case, any can be convinced of the original signer's agreement on the signed message from the proxy signature $(m_w, K_o, m, K, S, ASID)$.

Theorem 14. *Any can identify the actual proxy signers from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. From the proxy signature verification Equation (15), it can be seen that all actual proxy signers' identities and public keys are used. Therefore, any can identify the actual proxy signers from the proxy signature $(m_w, K_o, m, K, S, ASID)$.

Theorem 15. *Any can distinguish proxy signatures from normal signatures.*

Proof. In the proxy signature verification Equation (15), not only the original signer's public key, but also the actual proxy signers' public keys are used. In normal signature verification equation, only signers' public keys are used. So, any can distinguish proxy signatures from normal signatures.

Theorem 16. *The proxy signers can't repudiate having produced the proxy signature which has ever been signed to any one.*

Proof. As seen in proxy signature verification Equation (15), the actual proxy signers' identities and public keys are used. Thus he can't deny having produced the proxy signature which has ever been signed to any one.

From the above several theorems, we know that the proposed scheme can fulfill the securities of verifiability, strong identifiability, distinguishability, strong unforgeability, strong nonrepudiation, proxy protection and prevention of misuse of proxy signing power. In other words, the proposed scheme can resist equation attacks, collaboration attacks, public key substitution attacks, internal attacks and external attacks. In addition, the certificates of users are not needed in the proposed scheme. If users want to change his private key or the **CA** wants to change users' public key, both have to interact to finish it, or else neither of the two parties can succeed. The verifier only needs to obtain the public keys of the original signer and the proxy signers and needn't verify their validity as the verification of their public keys and the proxy signature is executed together. Thus, the self-certification of public keys can be realized.

5 Performance Evaluation

To facilitate the performance evaluation, we denote the following notations: T_h : The time for performing a one-way hash function h ; T_{mul} : The time for performing a modular multiplication computation; T_{add} : The time for performing a modular addition computation; T_{inv} : The time for performing a modular inverse computation; T_{pa} : The time for performing a point addition computation; T_{sm} : The time for performing a scalar multiplication computation; $|x|$: The bit-length of an integer x or a point x. The computational overhead and communication cost of the proposed scheme are stated in Table 1 and 2, respectively.

Table 1. Computational overhead of the proposed scheme

Phases	Computational overheads
Registration	User: $T_{pa} + 3T_{sm} + T_{add} + 2T_h$ The **CA**: $T_{pa} + T_{sm} + T_{mul} + T_{add} + T_h$
Proxy share generation	The original signer: $T_{sm} + 2T_{mul} + T_{add} + T_h$ Each proxy signer: $T_{pa} + 2T_{sm} + T_{mul} + T_{add} + 2T_h$
Proxy signature issuing	Each proxy signer: $(t'-1)T_{pa} + T_{sm} + 3T_{mul} + 2T_{add} + T_{inv} + T_h$ The clerk: $5t'T_{pa} + 7t'T_{sm} + (t'-1)T_{add} + t'T_{inv} + 4t'T_h$
Proxy signature verification	$(t'+3)T_{pa} + (t'+3)T_{sm} + (t'+2)T_h$

Table 2. Communication cost of the proposed scheme

Phases	Communication cost
Registration	$2\|G\| + \|n\| + \|ID_i\|$
Proxy share generation	$\|G\| + \|n\| + \|m_w\|$
Proxy signature issuing	$t'\|G\| + t'\|n\|$
Proxy signature verification	$2\|G\| + \|n\| + \|m_w\| + \|m\| + \|ASID\|$
Total [a]	$(t'+3)\|G\| + (t'+2)\|n\| + 2\|m_w\| + \|m\| + \|ASID\|$

[a] The total communication cost excludes the registration phase.

6 Conclusions

In the paper, based on Hsu et al's self-certified public key system and Hwang et al's proxy signature scheme, one ECDLP-based threshold proxy signature scheme with self-certified public key system and non Shamir's secret share protocol has been proposed. As far as we know, it is the first scheme based on ECDLP using self-certified public key system. The proposed scheme can provide needed security properties. In the proxy signature verification phase, the authentication of the original and the proxy signers' public keys and the verification of the threshold proxy signature are executed together. In addition, the computation overhead and communication cost of the proposed scheme are analyzed as well.

References

1. Mambo, M., Usuda, K., Okamoto, E.: Proxy Signature for Delegating Signing Operation. In: Proceedings of the 3th ACM Conference on Computer and Communications Security, pp. 48–57. ACM Press, New York (1996)
2. Li, J.G., Cao, Z.F.: Improvement of a Threshold Proxy Signature Scheme. J. of Computer Research and Development 39(11), 515–518 (2002)
3. Li, J.G., Cao, Z.F., Zhang, Y.C.: Improvement of M-U-O and K-P-W Proxy Signature Schemes. J. of Harbin Institute of Technology (New Series) 9(2), 145–148 (2002)
4. Li, J.G., Cao, Z.F., Zhang, Y.C.: Nonrepudiable Proxy Multi-signature Scheme. J. of Computer Science and Technology 18(3), 399–402 (2003)
5. Li, J.G., Cao, Z.F., Zhang, Y.C., Li, J.Z.: Cryptographic Analysis and Modification of Proxy Multi-signature Scheme. High Technology Letters 13(4), 1–5 (2003)
6. Hsu, C.L., Wu, T.S., Wu, T.C.: New Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. The J. of Systems and Software 58, 119–124 (2001)
7. Hwang, M.S., Lin, I.C., Lu Eric, J.L.: A Secure Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. International J. of Informatica 11(2), 1–8 (2000)
8. Hwang, S.J., Chen, C.C.: Cryptanalysis of Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Informatica 14(2), 205–212 (2003)
9. Sun, H.M.: An Efficient Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Computer Communications 22(8), 717–722 (1999)
10. Sun, H.M., Lee, N.Y., Hwang, T.: Threshold Proxy Signature. In: IEEE Proceedings on Ccomputers & Digital Techniques, pp. 259–263. IEEE Press, New York (1999)
11. Zhang, K.: Threshold Proxy Signature Schemes. In: Information Security Workshop, pp. 191–197 (1997)
12. Hsu, C.L., Wu, T.S., Wu, T.C.: Improvement of Threshold Proxy Signature Scheme. Applied Mathematics and Computation 136, 315–321 (2003)
13. Tsai, C.S., Tzeng, S.F., Hwang, M.S.: Improved Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Informatica 14(3), 393–402 (2003)
14. Hwang, S.J., Shi, C.H.: A Simple Multi-Proxy Signature Scheme. In: Proceeding of the Tenth National Conference on Information Security, Taiwan, pp. 134–138 (2000)
15. Denning, D.E.R.: Cryptography and Data Security. Addison-Wesley, Reading (1983)
16. Pedersen, T.: Distributed Provers with Applications to Undeniable Signatures, p. 547. Springer, New York (1991)
17. Li, L.H., Tzeng, S.F., Hwang, M.S.: Generalization of proxy signature-based on discrete logarithms. Computers & Security 22(3), 245–255 (2003)
18. Hsu, C.L., Wu, T.S.: Efficient proxy signature schemes using self-certified public keys. Applied Mathematics and Computation. In: Press, Corrected Proof, Available online July 9 (2003)

19. Hwang, M.S., Tzeng, S.F., Tsai, C.S.: Generalization of proxy signature based on elliptic curves. Computer Standards & Interfaces 26(2), 73–84 (2004)
20. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1991)
21. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
22. National Institute of Standards and Technology (NIST), The digital signature standard proposed by NIST. Communication of the ACM 35(7), 36–40 (1992)
23. Chang, C.C., Hwang, M.S.: Parallel computation of the generating keys for RSA cryptosystems. IEE Electronics Letters 32(15), 1365–1366 (1996)

Building Efficient Integrity Measurement and Attestation for Mobile Phone Platforms

Xinwen Zhang[1], Onur Acıiçmez[1], and Jean-Pierre Seifert[2]

[1] Samsung Information Systems America, San Jose, CA, USA
{xinwen.z,o.aciicmez}@samsung.com
[2] Deutsche Telekom Laboratories and Technical University of Berlin
jean-pierre.seifert@telekom.de

Abstract. Integrity measurement and attestation mechanisms have already been developed for PC and server platforms, however, porting these technologies directly on mobile and resource-limited devices does not truly satisfy their performance constraints. Therefore, there are ongoing research efforts on mobile-efficient integrity measurement and attestation mechanisms. In this paper we propose a simple and efficient solution for this problem by considering the unique features of mobile phone devices. Our customized secure boot mechanism ensures that a platform can boot to a secure state. During runtime an information flow–based integrity model is leveraged to maintain high integrity status of the system. Our solution satisfies identified security goals of integrity measurement and attestation. We have implemented our solution on a LiMo compatible mobile phone platform.

1 Introduction

Mobile devices such as cellular phones and smartphones have been evolved to be more open and general-purpose so that security has become a significant issue for device manufactures, network and service providers. On one side, today's smartphones typically have increasing processing power, integrated functions, and network connectivity, and hence, there are more services deployed on these devices not only voice but also data, such as messaging, content sharing, and enterprise data processing. On the other side, mobile phone devices face the same kind of threats that have been surging in PC world. According to F-Secure [22], currently there are more than 350 mobile malware in circulation. Examples of the most notorious threats to cellphones include Skull [16], Cabir [2], and Mabir [10] targeting at Symbian operating systems. McAfee's 2008 mobile security report [11] indicates that nearly 14% of global mobile users have been directly infected or have known someone who was infected by a mobile virus, and more than 70% of users expect mobile operators or device manufacturers to preload mobile security functionality.

One major attack mechanism of mobile malware is to maliciously change system files or configurations thus disable some phone or platform functions. For example, mobile viruses like Dampig [4], Fontal [6], Locknut [9], and Skulls [16] maliciously modify system files and configurations thus disable application manager and other legal applications on a phone. Phones can even become unable to uninstall or disable the malware once they are infected. Doomboot [5] installs corrupted system binaries into c: drive of a Symbian phone, and when the phone re-boots these corrupted binaries are loaded instead of the correct ones, and the phone crashes at boot. Similarly Skulls [16] can break phone services like messaging and camera. For another example, Cardblock [3] deletes system directories and destroys information about installed applications, MMS and SMS messages, phone numbers stored on the phone, and other critical system data. According to F-Secure [22], user installed applications are major threats to mobile platform security, including those downloaded from Internet or received from MMS and Bluetooth.

All these existing attacks compromise the integrity of a mobile phone device. Integrity measurement is a technique to enable a party to query the integrity status of software running on a platform, e.g., through attestation challenges. Several integrity measurement mechanisms have been proposed and developed. Trusted Computing Group (TCG) has published a set of specifications to measure, store, and report hardware and software integrity via transitive trust concept with hardware root-of-trust called Trusted Platform Module (TPM) and Core-Root-of-Trust-Measurement (CRTM). On a TPM-enabled platform, the CRTM measures the bootloader of the system before it is executed, and then stores the measured value into one of the platform configuration registers (PCRs) inside the TPM. The bootloader then loads OS image, measures it and stores via PCR extension, and then executes it [18]. In turn, the OS measures the loaded applications and stores their integrity values in PCRs before executing them. Upon an attestation challenge from a third party, the TPM signs a set of PCR values with an attestation identity key (AIK) and sends back the result. The challenger then can make decisions on the trust status of the platform by verifying the integrity of these values and comparing with the corresponding known-good values. For mobile phone platforms, TCG Mobile Phone Working Group has released specifications for mobile trusted module (MTM) [17].

Problems. IBM Integrity Measurement Architecture (IMA) [26] is an implementation of TCG specification in Linux. In IMA, GRUB is augmented to include measurement functions for kernel, ramdisk images, and grub.conf. Also, a kernel module is implemented to measure OS components after the kernel is loaded (post-boot), including libraries, usual command and tool binaries, configuration files, and application modules. Unfortunately, there is no practical implementation of integrity measurement and attestation for mobile platforms so far. One major reason is that, mobile devices such as cellular phones are still limited in computing power, e.g., they typically employ processors running at 200-600 MHz and internal RAM memories of 32MB or 64MB in size. This mandates that any security solution must be very efficient and leave only a tiny footprint in the limited memory. However, IMA has 400-600 measurements on typical desktop and server platforms [7], out of which around 350 measurements are post-boot measurements. The measurements take long time during boot and introduces significant performance delay during runtime. PRIMA [25] extends IMA on mobile phone devices and simplifies the integrity measurements. However it still has more than 200 measurements on an Openmoko phone device [25]. Although many software components (e.g., libs and binary images) on a phone are smaller than corresponding components in PC and servers, the performance overhead is still very significant and degrades the overall user experience.

Contributions. In this paper, we first identify the security goals and requirements for mobile platform integrity measurement. We then propose an efficient solution towards these goals and requirements. Our major design objective is to reduce the number of software components to be measured on a platform, and thus reduces booting and runtime performance overhead and eases the integrity verification for attestation. Our solution consists of two major steps. First, we develop a secure boot mechanism to ensure that a secure kernel is booted, which in turn ensures a well-behaved measurement agent and enforces an authenticated security policy during runtime of the operating system (OS). Runtime integrity assurance is then achieved by leveraging a simple integrity model, which efficiently identifies the *trusted* and *untrusted domains* (including processes and resources) of a mobile platform, and restricts modifications from untrusted domain to trusted side. The essential idea of our integrity measurement is that we only measure integrity protection policy and enforcement mechanism based on our model, including OS kernel and user space components that accept inputs from untrusted domain on a platform (mainly framework daemons providing services to other applications on a platform). For verification, if our model is correctly enforced with carefully designed policy, we have the assurance that processes and resources in trusted domain are protected from untrusted applications, and the platform is always in good integrity status during runtime. The rational that makes our approach practical is that in mobile phone and many embedded devices, it's much easier to identify the board line between trusted and untrusted domains, as we discuss in later sections of this paper. While in PC and server environments, due to many network-faced services and installed software from many unverified resources, it's a complex task.

Outline. In the next section we analyze the general security goals of integrity measurement and attestation for mobile platform. We then present our solution with secure boot and runtime integrity model to satisfy these goals in Section 3. We briefly present some implementation information of our approach on a LiMo compatible real smartphone device in Section 4. Section 5 presents some related work on platform integrity measurement. We conclude this paper in Section 6.

2 Security Goals

In this section we explain the security goals of integrity measurement and attestation on general computing environments. We then identify some further requirements for mobile platforms. We note that this is not a complete list of security goals, but the aspects considered in our solution.

Trusted Boot. TCG specification requires trusted boot, which adopts the concept similar to AEGIS [19]. Specifically, upon booting, all software components including BIOS and OS bootloader are transitively measured by the CRTM of a platform and the results are extended in the PCRs of TPM. The bootloader then loads OS image, measures it and stores via PCR extension, and then executes it [18]. In turn, the OS can measure applications and stores their integrity values in PCRs before executing them.

Load-time Integrity Measurement. There are two different types of integrity measurement for a software binary: load-time and runtime measurements. The TCG only specifies load-time integrity measurement–a piece of code or data is measured when it is mapped or loaded into main memory. In IMA [26], measurements are invoked in several system call functions such as `mmap` and `insmod` when code or kernel modules are loaded but before they are executed. After a code is mapped into memory and during runtime, it is very difficult to measure the integrity of the process considering very dynamic and undeterministic behaviors of typical applications, such as loading active code, receiving external inputs, and allocating dynamic memory.

In addition to trusted boot and load-time integrity measurement, integrity protection for mobile phones has the following extra requirements.

Secure Boot. Due to the business model of mobile industry, a mobile device is a service convergence of many remote service stakeholders, including device manufacturer, network operator, and other service providers. There is a set of *mandatory engines* [17] reside on a single mobile platform and provide critical and indispensable services, which have to be running in known-good states, which mean that the integrity of these services must be verified to assure their trustworthiness. Therefore TCG Mobile Phone Reference Architecture [17] states that secure boot is mandatory for MTM. As mentioned, trusted boot requires any component during boot has to be measured and the result is securely stored. Further, secure boot requires that a measured value has to be verified before it is executed. If the measured result does not match a known-good value, the booting is aborted.

Low Booting and Runtime Overhead. Most mobile devices such as cellular phones and smartphones are still limited in computing power. Also, user experience is a key business factor in consumer electronic (CE) market. This requires any security solution to be very efficient not to degrade overall user experience. For example, a typical mobile phone should boot in a reasonable time (e.g., less than 10s) and should not have a high overhead when starting a widget or opening the keyboard screen when making a phone call. Therefore low booting and runtime overhead is a pressing requirement for mobile platforms and integrity measurement during boot and in post-boot state should not degrade the performance and user experience too much.

Runtime Integrity Assurance. Although runtime integrity measurement is not practical in both PC and mobile platforms, there should be some mechanism to preserve the integrity level of critical applications and resources during runtime, e.g., phone related services (telephony server) and platform management agents. Both TCG and IMA do not propose any mechanism for this purpose.

3 Our Approach

Overview. We leverage two mechanisms to achieve these security goals. First of them is our security enhanced bootloader, which measures the kernel image, software TPM module, and our security policy file when the platform boots up. If all these are authenticated, we can assure that the base system (i.e., kernel) is trusted to (1) carry out the rest of the integrity measurements and (2) enforce the security policy after booting. Our second mechanism relies on a simple integrity model (crafted specificly for mobile platforms) to identify the boundary between trusted and untrusted domains on a typical mobile platform and control the information flow. Security policy implementing our integrity model is enforced in both kernel and user space. The result is that only a few number of binaries and data objects need to be measured in post-boot phase, as they enforce our policy and handle service requests from both trusted and untrusted applications, e.g., most service daemons for platform management, telephony service, and other trusted third party services.

Security Assumptions. We do not consider attacks in kernel, such as installing kernel rootkits[1] to bypass our security policy enforcement in kernel space. We further assume that an attacker cannot re-flash the bootloader on a mobile platform, e.g., which can be stored in ROM. That is, our integrity measurement and attestation ensures that a system is secure against software-based attacks, which is the major objective of trusted computing technologies [18].

3.1 Secure Boot

The challenge for secure boot is that there is no hardware implementation of TPM or MTM available for mobile phones so far. Our secure boot leverages a measurement agent in bootloader. For security considerations, we assume that the bootloader and especially this agent is tightly coupled with and protected by hardware, e.g., can be written into ROM. Moreover, a public RSA key of the device manufacturer is encoded in the measurement agent. We assume that the following known-good integrity values (SHA1 value) of the kernel image, software TPM module (swTPM), and security policy are signed by the device manufacturer and the corresponding certificates are stored along with these binaries in the platform. As an alternative, these values can be received from a trusted service provider over-the-air, e.g., from a server and signed by the mobile network operator of the device, in which case the credentials would include the public key certificate of the operator. This enhances the deployment flexibility as the kernel and security policy can be updated after the device is released to customer.

When the platform boots, the measurement agent measures the integrity of the kernel image, the swTPM module, and the binary policy and verifies these values by comparing with those in the certificates. The kernel is executed only after a success verification. The measured integrity values of the kernel image, swTPM, and the policy file are passed to the kernel upon start of its execution. The kernel loads the policy file and starts enforcing the access control rules following our integrity model. Recall that in our architecture, we rely on kernel level mandatory access control mechanism, i.e., the kernel has the ability to fully control the entire information flow in the platform. The kernel also initiates the swTPM and extends the PCRs using the the integrity parameters passed from the bootloader. Through these steps, we ensure that a secure kernel

[1] According to F-Secure [22], rootkits have not been found on cellphone devices.

is loaded, an authenticated swTPM is working, and the kernel is trusted to enforce an authenticated security policy during runtime.

3.2 Secure Runtime

Due to very dynamic and undeterministic behaviors of typical applications, runtime integrity measurement is not practical in typical OS environment, if not impossible at all. We propose to leverage an integrity model for runtime integrity preservation. The fundamental idea is, if needed, a trusted application always receives information from other trusted applications or reads data from trusted domains, and untrusted processes cannot write to trusted resources. Through this, the integrity of trusted services and resources is preserved. That is, our model is a runtime information flow control mechanism.

Our model is built on the unique integrity requirements of mobile platforms. Typically, for desktop and server platforms, one of the major security objectives is to protect network-faced applications and services such as httpd, smtpd, ftpd, and samba, which accept unverified inputs from others [23]. As mentioned in Section 1, for mobile phone platforms, on the other side, the major security objective is to protect system integrity threaten by user installed applications, including those downloaded from Internet and received from MMS and Bluetooth.

The architecture of a mobile platform is different from conventional desktop systems. For example, LiMo platform includes a set of frameworks, which provide services to applications via function interfaces. Applications running on a mainstream Linux OS in (e.g.) a desktop communicate directly with kernel to access hardware resources through system call APIs. On the other hand, frameworks control the access and usage of hardware resources on mobile phone devices. For example, mobile phone applications can access SIM data only through the interface APIs of the telephony framework. This architecture is heavily dependent on the business model of mobile and wireless telecommunication industry, where different stakeholders or vendors (device manufactures, network carriers, and third-party service providers) provide variant frameworks and give interfaces to application developers.

A trusted process such as a service daemon may accept requests from both trusted and untrusted applications concurrently. Traditional integrity models are not efficient and flexible under these scenarios. For example, LOMAC [20], UMIP [23], and CW-lite [25] require a process dynamically downgrade its security level whenever it accesses low integrity objects or receives inputs from low integrity processes. However, the process needs to re-start whenever it needs to access high integrity objects later, which is not efficient for mobile devices. Therefore, we propose our integrity model as follows.

Integrity Model. Like traditional security models, our model distinguishes subjects and objects in OS. Basically, subjects are *active entities* that can access to objects, which are *passive entities* in a system such as files and sockets. In our model, the set of subjects S are mainly active processes and daemons, the set of objects O include all possible entities that can be accessed by processes, and $S \subseteq O$. In an OS environment, there are many different types of access operations. In our model, for integrity purposes, we focus on three access operations: create, read, and write. From information flow perspective, all other operations can be mapped to these three operations [15].

Table [1] shows the basic integrity rules in our model, where $L(o)$ is the integrity level of object o. Note that by default we consider trusted entities as high integrity and untrusted entities as low integrity in this paper. The first two creating rules are *exclusively* applied upon a single object creation. Typically, the first rule applies to objects that are *privately* created by a process. For example, an application's logs, intermediate and output files are private data of this process. The second rule applies to objects that are *precatively* created by a process upon the request of another process. In one case, $s1$ is a server running as a daemon process, the $s2$ can be any process that leverages the function of the daemon process to create objects, e.g., to create a GPRS session, or save its configuration data with GConf daemon (gconfd). In another case, $s1$ is a common tool or facility program that can be used by $s2$ to create object. In these cases, the integrity level of the created object is corresponding to that of $s2$.

Table 1. Integrity Rules

Policy Rule	Description
r1: $create(s, o)$: $L(o) = L(s)$	object o is created by a process s.
r2: $create(s1, s2, o)$: $L(o) = MIN((L(s1), L(s2))$	object o is created by process $s1$ with input from another process $s2$
r3: $can\_read(s, o)$: $L(s) \leq L(o)$	a process s only can read from an equal or higher integrity process or object o.
r4: $can\_write(s, o)$: $L(s) \geq L(o)$	a high integrity process s can write to an equal or lower integrity process or object o
r5: $can\_read(s1, s2, o)$: $L(s1) \geq L(s2) \wedge can\_write(s1, o) \wedge L(s2) \geq L(o)$	a high integrity process $s1$ can receive information from an equal or lower integrity subject $s2$, provided that the information will be written to lower integrity subject or object o by the high integrity process $s1$.

The r3 and r4 rules indicate that there is no restriction on information flow within trusted entities, and within untrusted entities, respectively, which is, fundamentally, a BIBA-like integrity policy. Rule r5 states that a trusted subject to behave as a communication or service channel between untrusted entities. Therefore not every subject can be trusted for this purpose. Typically, communications between applications and service daemons can be modelled with this rule. For example, on a mobile phone device, a telephony daemon can create a voice conversation between an application and wireless modem, upon the calling of telephony APIs. A low integrity process cannot modify any information of the connection created by a high integrity process, thus preventing stealthily forwarding the conversation to a malicious host, or making the conversation into a conference call.

Boundary of Trusted and Untrusted Domains. Many embedded devices (including mobile phones) use read-only filesystems to store static binary images and data for system and application software, which is not a typical characteristics of desktop and server platforms. On a LiMo compatible smartphone [8] that we have evaluated, all Linux system binaries (e.g., `init`, `busybox`), shared libraries (`/lib`, `/usr/lib`), scripts (e.g., `inetd`, `network`, `portmap`), and non-mutable configuration files (`fstab.conf`, `inetd.conf`, `inittab.conf`,`mdev.conf`) are located in a read-only cramfs filesystem. Also, all phone related application binaries, configurations, and framework libraries are located in another cramfs filesystem. All mutable phone related files are located in an ext3 filesystem, including logs, tmp files, database files, GConf configuration resource files, and user-customizable configuration files (e.g., application settings and GUI themes). All user applications can only be installed in a dedicated directory of the ext3 filesystem and the `/mnt/mmc`, which is mounted when a flash memory card is inserted. Based on this layout, we decide that all objects in user writable filesystem and directories are untrusted, and the others (including read-only and read-write filesystems and directories) are trusted. This is based on the integrity objective of our solution – to protect system and service components from user installed applications. We have observed very similar filesystem layout on many other Linux mobile phones, including Motorola A1200 [14] and Android [1].

We note that above approach to determine trusted and untrusted domains is just an example mechanism. The general idea of our model is to decide the boundary between platform system and service domains, and user installed application domains, via different filesystems and/or directories. Note that, filesystem layout is determined by the device manufacturer of a platform and such a separation can be enforced quite easily. This approach simplifies policy definitions and reduces runtime overhead compared to traditional approaches such as SELinux on PCs, which sets the trust boundaries based on individual files.

Integrity-aware IPC. Our integrity-aware IPC aims to control information flow between subjects. Most IPC objects (including domain sockets, pipes, fifo, message queues, shared memory, and shared files) inherit their integrity levels from the processes that create them. Therefore, our model restricts any access where a high integrity process tries to receive data through an IPC object created by a low integrity process.

In many mobile Linux platforms such as LiMo, OpenMoko, GPE, Maemo, and Qtopia, D-Bus is the major IPC, which is a message-based communication mechanism between processes. A process builds a connection with a system- or user-wide D-Bus daemon (dbusd). When this process wants to communicate with another process, it sends messages to dbusd via this connection. The dbusd maintains all such connections (from many different processes), and routes messages between them. A D-Bus message is an object in our model, which inherits integrity level from its creating process. According to r5 of our model, trusted subject dbusd can receive any D-Bus message (low or high integrity level) and forward to corresponding destination process. Typically, a trusted process can only receive high integrity messages from dbusd. According to r5 of our model, if a process is a trusted daemon, like telephony service or GConf daemon, it can receive high and low integrity messages from dbusd, and handle them separately within the daemon.

Program Installation and Launching. Application installation is usually performed by a dedicated installation program called installer. As the installer is trusted, it can read both high and low integrity application packages according to our model. Again according to our model, it can write (i.e., install) to trusted part of the filesystem if it reads the data from a high integrity software package, and can write to untrusted part of the filesystem if the data is from a low integrity software package. Similar to installation, during the runtime of a mobile system, a process is invoked by a trusted program called program launcher, and both high and low integrity processes can be invoked by the program launcher. All processes invoked from trusted program files are in high integrity level, and all processes invoked from untrusted program files are in low integrity level.

Mobile phones can be infected with malware via variant communication channels between phone devices and networks. For example, many malware in Symbian-based phones send malicious codes via Bluetooth channel, or distribute with MMS messages (either in message contents or as attachments). Therefore in our model we treat any code received from these network-faced applications as untrusted by default. Thus, according to our model, any process invoked from arbitrary code by MMS agent or browser is in low integrity level and cannot write to trusted resources and services, such as corrupting system binaries or changing platform configurations. It is possible that some software received from Bluetooth/MMS/browser can be trusted. For instance, a user can download a trusted software from his PC via Bluetooth or from a trusted service provider's website via browser. Therefore it can be installed to the trusted side on read-write filesystem by the application manager on the phone, after the user explicitly indicates or confirms that this application is trusted, or with a source verification (e.g., via digital signature) of the software package.

3.3 Putting Together: Integrity Measurement and Protection

After introducing our secure boot mechanism and runtime integrity model, it is time explain how to measure and verify a mobile platform integrity. The overall platform architecture and measurements are show in Figure 1. As aforementioned, with secure boot, a mobile platform can boot to a state with an authenticated kernel, which in turn is trusted to enforce an authenticated policy based on our model. The integrity values of kernel image, swTPM and the policy are stored in PCRs of the swTPM, e.g., for attestation purposes after booting. After this, a measurement agent in the kernel and the swTPM can perform secure measurements and storage for other modules and trusted subjects during runtime. Specifically, when a measurement target is to be loaded, the measurement agent captures this event, measures its integrity, extends a corresponding PCR of the swTPM, and then loads the target into main memory and transfers its execution to it.

During runtime, our objective is to minimize the number of components to be measured for performance reasons. Firstly, according to our integrity model, when the kernel is genuine, we have the assurance that all code and data in read-only filesystems cannot be altered by any means. Secondly, for any code that is located in a read-write filesystem and regarded as trusted, we check whether the code is an executable binary of a daemon, which is a trusted subject and enforces security policy in our model. If it is, we need to measure it. If the code is not a trusted daemon, we do not need to measure it, as its runtime integrity is preserved with the integrity policy rules based on our model, and the policy is honestly enforced by the kernel and daemons. We do not need to measure any other untrusted code and data, because they are already regarded as untrusted subjects and objects in our model and cannot alter any trusted entities due to our model and policy enforcement mechanisms explained above. Similar to IBM IMA [26], we do not measure dynamic data and configuration files.

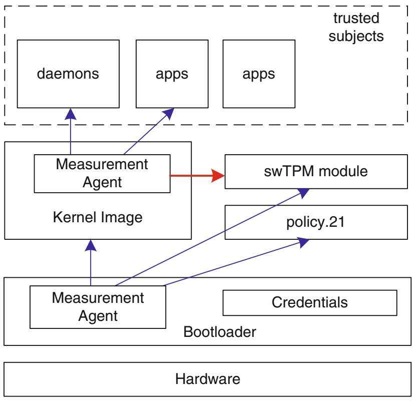


Fig. 1. Platform architecture: integrity measurement and verification with secure boot and runtime integrity enforcement

The attestation of our platform is similar to traditional TCG-based approach. With an attestation service agent on the platform (not shown in Figure 1), when receiving an attestation challenge, the agent responses with PCR values signed by the swTPM. Due to space limitation, we ignore the details of these steps and also the procedure to obtain an attestation identity key for the platform.

4 Implementation

We have implemented our model on a real LiMo platform [8]. We augmented the bootloader (u-boot) on this platform with a simple integrity measurement agent. We ported the TPM emulator [27] on this platform. We are also porting the software MTM module [12] to this platform.

The integrity model is implemented with SELinux, which provides comprehensive security checks via Linux Security Module (LSM) in kernel. SELinux provides domain-type and role-based policy specifications, which can be used to define policy rules to implement high level security models. Actually, current SELinux does not have an integrity model built-in. On one side, our implementation simplifies SELinux policy for mobile phone devices by leveraging trusted and untrusted subject attributes. On the other side, our implementation augments SELinux policy with our integrity model.

D-Bus is the major IPC mechanism for most Linux-based mobile platforms. Following our integrity model, we extend D-Bus implementation in two aspects. First, each message is augmented with a header field to specify its integrity level based on the process which sends the message, and the value of this field is set by dbusd when it receives the message and before dispatches it. Secondly, according to our integrity model, if a destination bus name is a trusted daemon process, it can accept both high and low integrity messages; otherwise, it only accepts messages with the same integrity level as itself. We also have augmented other framework daemons of this LiMo platform to enforce similar integrity policy, including the GConf and telephony server, as they can access both trusted and untrusted messages via D-Bus.

We define two domain attributes to specify high and low integrity processes: *trusted* and *untrusted*, respectively. We use generalized filesystem labeling mechanism [13] in SELinux to label both cramfs and ext3 filesystems. Our policy size is less than 20KB including `genfscon` rules for filesystem labelling. Comparing to that in typical desktop Linux distributions such as Fedora Core 6 (which has 1.2MB policy file), our policy footprint is tiny. We also studied the performance of our runtime security enforcement with micro benchmark. Our results show that for most operations, our security enforcement has less than 4% overhead, which is significantly less than the counterpart technology on PC [24].

5 Related Work

As aforementioned, IBM IMA [26,7] is the integrity measurement solution for PC platforms. However, directly porting this to mobile devices is not practical due to its high computation overhead during booting and runtime. PRIMA [25] leverages the CW-lite information flow control to maintain a process's integrity, where particular interfaces of the process filter low integrity information when received by this process. However, identifying filtering interfaces in many service processes (daemons) on a mobile phone is not a easy task, especially many of them come from different software vendors, e.g., network carrier, device manufacturer, and third party service providers. Also, PRIMA still has more than 200 measurements on an Openmoko phone device [25].

Dynamic root-of-trust-for-measurement (DRTM) [21] is a mechanism on x86 platform to execute a piece of code in hardware protected trusted environment during runtime and without physically rebooting the platform. However this is not practical in mobile device. First of all, ARM processors have not built in this capability. Secondly, only 32KB size of code can be executed in the trusted environment in DRTM, which shifts the integrity measurement problem to other mechanism after that code is launched.

Runtime integrity measurement has been proposed recently with copy-on-write mechanism [28]. However, this is implemented on Xen-like virtualization environment, which is so far not practical for mobile platforms.

6 Conclusion

Towards the protection of mobile platform integrity threaten from untrusted user applications, we propose a simple and efficient solution for integrity measurement and attestation. Our solution uses a secure bootloader to measure the kernel and a software TPM, which ensures that the platform can boot to a secure state. After booting, we leverage a simple integrity model to preserve the runtime integrity of the system. Our model easily distinguishes trusted and untrusted domains on both filesytem and memory space, thus making the policy development very simple and verifiable. During runtime, we measure the enforcement mechanism of security policy based on our integrity model, i.e., trusted daemon processes, thus significantly reduce the number of components to be measured. We are developing a formal specification and security property verification of our proposed integrity model.

References

1. Android, http://code.google.com/android/
2. Cabir, http://www.f-secure.com/v-descs/cabir.shtml
3. Cardblock, http://www.f-secure.com/v-descs/cardblock_a.shtml
4. Dampig, http://www.f-secure.com/v-descs/dampig_a.shtml
5. Doomboot, http://www.f-secure.com/v-descs/doomboot_a.shtml
6. Fontal, http://www.f-secure.com/v-descs/fontal_a.shtml
7. IBM integrity measurement architecture, http://domino.research.ibm.com/comm/research_projects.nsf/pages/ssd_ima.index.html
8. Limo Foundation, http://www.limofoundation.org/en/technical-documents.html
9. Locknut, http://www.f-secure.com/v-descs/locknut_e.shtml
10. Mabir, http://www.f-secure.com/v-descs/mabir.shtml
11. Mcafee Mobile Security Report (2008), http://www.mcafee.com/us/research/mobile_security_report_2008.html
12. MTM Emulator, http://hemviken.fi/mtm/
13. NSA Security-Enhanced Linux Example Policy, http://www.nsa.gov/selinux/
14. OpenEZX, http://wiki.openezx.org/main_page
15. Setools–policy analysis tools for selinux, http://oss.tresys.com/projects/setools
16. Skulls, http://www.f-secure.com/v-descs/skulls.shtml
17. TCG Mobile Reference Architecture Specification Version 1.0, https://www.trustedcomputinggroup.org/specs/mobilephone/tcg-mobile-reference-architecture-1.0.pdf
18. TCG TPM Main Part 1 Design Principles Specification Version 1.2, https://www.trustedcomputinggroup.org
19. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A secure and reliable bootstrap architecture. In: Proc. of IEEE Conference on Security and Privacy, pp. 65–71 (1997)
20. Fraser, T.: LOMAC: MAC you can live with. In: Proc. of the 2001 Usenix Annual Technical Conference (2001)
21. Grawrock, D.: The Intel Safer Computing Initiative: Building Blocks for Trusted Computing. Intel Press (2006)
22. Hypponen, M.: State of cell phone malware in 2007 (2007), http://www.usenix.org/events/sec07/tech/hypponen.pdf
23. Li, N., Mao, Z., Chen, H.: Usable mandatory integrity protections for operating systems. In: Proc. of IEEE Symposium on Security and Privacy (2007)
24. Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. In: Proc. of USENIX Annual Technical Conference, June 25-30, pp. 29–42 (2001)
25. Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring integrity on mobile phone systems. In: Proc. of the 13th ACM Symposium on Access Control Models and Technologies (2008)
26. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: USENIX Security Symposium (2004)
27. Strasser, M.: Software-based TPM emulator for linux. Semester Thesis, Department of Computer Science, Swiss Federal Institute of Technology Zurich (2004)
28. Thober, M., Pendergrass, J.A., McDonell, C.D.: Improving coherency of runtime integrity measurement. In: Proc. of the 3rd ACM workshop on Scalable Trusted Computing (2008)

Context-Aware Monitoring of Untrusted Mobile Applications

Andrew Brown and Mark Ryan

School of Computer Science, University of Birmingham, B15 2TT, UK
{A.J.Brown,M.D.Ryan}@cs.bham.ac.uk

Abstract. Current measures to enhance the security of untrusted mobile applications require a user to trust the software vendor. They do not guarantee complete protection against the behaviours that mobile malware commonly exhibits. This paper expands *execution monitoring*, building a more precise system to prevent mobile applications deviating from their intended functions. User judgements about program execution can be specified abstractly and compiled into a monitor capable of identifying an event's context. We demonstrate our development of a prototype system for the BlackBerry platform and show how it can defend the device against unseen malware more effectively than existing security tools.

1 Introduction

The number of mobile devices in operation today exceeds the size of the population in over 30 countries. Of these, roughly 30% are "smart" devices that allow the user to download, install and execute third-party applications. Such an expansion in device capabilities has made them more vulnerable to attack. This paper focuses on the prevention of *unseen mobile malware*, distributed by a malicious third-party in order to compromise the confidentiality, integrity and availability of a user's interaction with mobile data and services.

1.1 Mobile Malware Defence

Java Micro Edition (Java ME) is the most popular framework for executing third-party mobile applications. It consists of two components: the *Connection Limited Devices Configuration* (CLDC) [11], which facilitates program access to the native methods which control device hardware, and the *Mobile Information Device Profile* (MIDP) [12], which provides programmers with abstractions of generic device functionalities (e.g., sending SMS and e-mail messages). Java ME applications are called MIDlets.

Java ME's security model is not flexible enough to allow guarantees to be made about all security-relevant events an application might perform. In most cases, it only takes into account the application provider (i.e., the signatory of the MIDlet). Java Security Domains [14] are implemented in some architectures to block security-relevant API calls, though this can also prevent an application being used for its primary purpose.

Application-level virtualisation allows users to run additional security tools to verify the identity and integrity of the downloaded application. *Code signing*, *discretionary access controls* and *signature-based anti-virus software* are the most commonly used approaches. Each has the following associated problems:

– *Digitally signing* a hash of an executable can confirm its author and guarantee that it has not been altered since it was signed, but does not guarantee the quality or security of code the application will execute.
– *Access controls* contribute to a systematic security framework[1] but can be inflexible, with default settings leaving the device vulnerable to attack and stricter ones impeding program functionality.
– *Mobile anti-virus* software requires signature dictionaries to be downloaded and updated, which is infeasible for devices with low power and/or limited network connectivity. Further, attack recovery is rather rigid; it simply deletes executable files.

1.2 Execution Monitoring

We employ a well-known technique called *execution monitoring* [1,3,5,7,8,9,16,17,18] to defend devices against malicious and defective software. An execution monitor is a co-routine that runs in parallel with a target program, fully regulating its interaction with its host machine. The approach can prevent and recover from harmful behaviour in real-time, rather than after an attack has succeeded, with minimal disturbance to legitimate program features. Execution monitors analyse either system calls or API calls, providing their host with a more *fine-grained* view of target application behaviour.

Despite these advantages, execution monitoring has not been widely adopted by developers building anti-malware products. The following reasons explain this:

Specifying an execution monitor is a complex process. Monitor implementation requires sequences of triggering and recovery events to be defined. High-level policy languages exist to counteract this problem, but most require reasoning about misuse event sequences in an imperative fashion and so are difficult for non-technical users comprehend.

Execution monitors cannot precisely capture malicious behaviour patterns. Identification of the *context* in which a target program invokes an event is required to achieve this. Existing monitoring schemes, including those for Java ME [5], cannot store sufficient information about an event's properties and so restrict application behaviour beyond the set of security-relevant events.

[1] Access controls can prevent an unsigned application being linked to the CLDC APIs which control native device functions. This type of mitigation is often still too coarse-grained.

1.3 Paper Structure

Section 2 presents our approach to allow more fine-grained control of mobile applications. We develop declarative language called Application Behaviour Modelling Language (ABML) to express policies that identify the context of a program event. In Section 3, we use our operational experiences in mitigating BlackBerry intrusions to demonstrate our work and provide an empirical evaluation. Section 4 details our enforcement mechanism for ABML, which works by translating a policy into Java source code, for input to a policy enforcement engine called Polymer [3]. We conclude by relating our contributions to those of others and proposing future work.

2 Modelling Application Behaviour

Mobile malware often uses HTTP, HTTPS, or other popular protocols to compose an attack. By employing seemingly 'legitimate' combinations of events and protocols to transmit sensitive information, unseen malware can avoid detection. Context-aware monitoring of event invocations can mitigate such attacks and preserve program functionality.

2.1 Approach Summary

Our approach places application monitoring wrappers around arbitrarily downloaded MIDlets in order to implement a security policy and so constrain undesirable functionality. We use these to apply user-specified policies in an aspect-oriented manner, creating flexible security against malicious components. Our policies are generic first-class objects and can be applied to any MIDlet as follows:

1. At download-time, a MIDlet's bytecode is scanned to identify all API calls it makes. Only application-relevant calls can be contained in a policy.
2. At specification-time, every high-level policy is compiled into monitor source code, which is contained within a Java class.
3. At load-time, every API call in the libraries a MIDlet can access is instrumented with calls to the monitor.
4. At runtime, control is passed to the monitor when any event which the policy reasons about is invoked.

Figure 1 illustrates the technologies we employ to implement our framework. Our approach uses Polymer[2] [3], a fully-implemented engine for enforcing execution monitoring threads on arbitrary Java programs. It responds to a policy violation by transforming a target to adhere to conditions of the applied policy.

[2] We have modified Polymer to allow it to operate on MIDlets, using MIDP2 and CLDC libraries [11,12].

Our system architecture targets two classes of user separately:

– *Administrators* can construct and distribute groups of policies to the devices which they manage, specifying them textually or by means of a comprehensive user interface.
– *Users* can manage and modify policies set by an administrator[3], or select a pre-specified policy from a repository, based on the class of program they perceive to have downloaded. Our MIDlet policy classes are *editor*, a *browser*, a *shell*, a *viewer*, a *transformer*, a *game*, and a *messenger*.

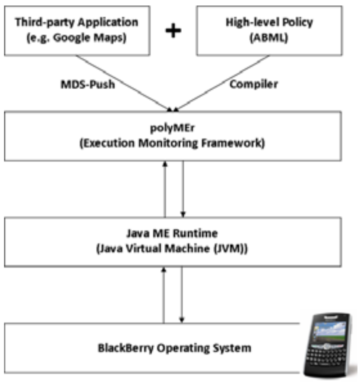

Third-party Application (e.g. Google Maps)
+
High-level Policy (ABML)
MDS-Push
Compiler
polyMEr (Execution Monitoring Framework)
Java ME Runtime (Java Virtual Machine (JVM))
BlackBerry Operating System

Fig. 1. Integration of Polymer and Java ME on the BlackBerry

2.2 Application Behaviour Modelling Language (ABML)

ABML is a policy specification language for reasoning about application behaviour in a declarative manner using arbitrary events. It is inspired by Behaviour Modelling Specification Language (BMSL) [17], in which event-based security-relevant properties can be expressed in order to capture the *intended* behaviour of a system, or *misuse* behaviours associated with vulnerabilities and their exploitation. ABML builds on BMSL with:

– High-level constructs for specifying event *histories* and *recovery sequences*;
– The notion and *strongly*- and *weakly*-ordered events;
– *Abstraction mechanisms* for referring to events *and* the data they use;
– Operators to increase the power of comparison between event conditions.

ABML's primary constructs are:

1. **Policies:** A policy consists of a set of rules, or a set of variables followed by a set of rules. Variable declaration may be *local* or *global*: a local variable's scope is limited to a single rule, whereas a global variable can be applied within all rules in a policy.
2. **Rules:** A rule is of the form $\mathsf{H} \rightarrow \mathsf{R}$, where H denotes a *history of actions* invoked by the target application which triggered a rule, and R denotes some postcondition which (i) either must hold true for that history, or (ii) must specify some responsive steps to be followed where a rule is triggered.
3. **History patterns:** A history, H, models the target application's instruction stream and is expressed by a pattern over a sequence of events: pat. A pat can measure event *occurrence*, *non-occurrence*, *sequencing*, *alternation* and *repetition*, providing a means of specifying the temporal properties of application behaviour [17].

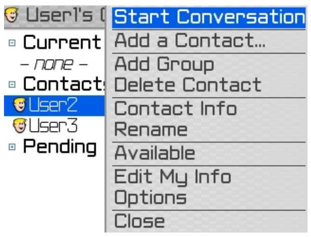
[3] Where the administrator has granted the user sufficient access permissions.

   (a) **Strongly-ordered patterns:** A strongly-ordered pat considers events strictly in the order they occur in. It is specified by separating events with a semi-colon (';').
   (b) **Weakly-ordered patterns:** A weakly-ordered pat allows events which a policy does not reason about to occur in between those events which it does reason about. This representation is suitable where events may happen simultaneously, or housekeeping operations are to be performed between events. It is specified by separating events with a double semi-colon (';;').

4. **Events:** The following may be referred to within a rule:
   (a) **Atomic events** have the form $\mathsf{e}(\mathsf{value}_1, \mathsf{value}_2, ..., \mathsf{value}_n)|\mathsf{cond}$, where e is an API call name, value is an argument to that call and cond is a boolean-valued expression on that call's arguments. We add comparison operators such as contains, startsWith and endsWith to enable more advanced reasoning about event conditions.
   (b) **Abstract events**, E, provide a means of concatenating atomic events to form suitable abstractions that reflect a host's ontology (e.g., *read from a local file*, *send an SMS message over carrier network*, *read data from device's GPS module*). An E is referenced using the constructs zone, resource and action, which allow one to refer to the zone a program may interact with (e.g., internet, network, localdevice), the resources within *that* zone it can access and the actions it may perform on *those* resources.
   (c) **Recovery events** allow users to reason about policy violations without having in-depth knowledge of the target application. A postcondition, R, assists this reasoning process. It allows the policy author to:
      – Deny the invocation of a triggering event ($\perp$);
      – Enforce a condition on a triggering event (cond):
         – If cond is *true* when a rule is evaluated, program execution is permitted to continue;
         – Otherwise, a sequence of recovery events of type E are derived from cond at compile-time.
      – Refer to an alternative event, which may be abstract (E);
      – Make a call to terminate the target application (`halt`).

5. **Event data usage:** An event's context is normally specified at the atomic level by populating the arguments to that event (e.g., setting one or more values in $\mathsf{e}(\mathsf{value}_1, \mathsf{value}_2, ..., \mathsf{value}_n)$). Because abstract events are collections of atoms, each may have a different argument signature. ABML therefore provides an event data tracking construct, which allows some abstract data object, O, to be instantiated at runtime with the concrete data used by that event. For example, HttpResponse represents a response over the HTTP protocol. It contains instance variables to capture that response's `head` (e.g., status code, date, server, content-type), `body` (e.g., (X)HTML, XML) and a reference to the HttpRequest which invoked it. Section 4.2 illustrates this process' enforcement.

3 Demonstration



In order to demonstrate ABML policy enforcement, we present our operational experiences in monitoring the execution of an untrusted MIDlets on the BlackBerry 8800 series device [10]. The MIDlet we consider is an instant messaging application, which was downloaded to the device over GPRS at the user's request. The user wishes to control the MIDlet's features with the device's input apparatus; initiate 'live' messaging sessions with remote devices in a contacts list; send asynchronous messages (e.g., SMS, e-mail) to a contact; and, receive software updates from a remote server.

3.1 Existing Device Protection

The MIDlet may invoke other behaviours which the user is unaware of, some of which may occur despite the BlackBerry's code signing or access control mechanisms. It must be linked to the CLDC libraries in order to execute and so must be signed. The user can set the device's access controls, as illustrated in Table 1, to prevent the signed code behaving maliciously.

Table 1. Settings for device access controls to prevent a MIDlet forwarding data

Attack on...	**Application Permission**	**Set to...**
1 – SMS	Connections > Carrier Internet	Deny
2 – E-mail	User data > E-mail	Deny
3 – TCP/IP	Connections > Carrier Internet	Deny
4 – Bluetooth	Connections > Bluetooth	Deny

Such measures clearly inhibit the primary functions of a messenger. For example, the only means the user has to prevent the MIDlet intercepting and forwarding SMS messages to an attacker is to deny its use of the BlackBerry Internet Service. This renders it unusable for the purpose for which the user downloaded it.

3.2 Attacks and Countermeasures

Users can apply an ABML policy which predicts and defends against attacks in a more fine-grained way. Such a policy would be crafted by an administrator or device vendor and applied by the user when they first download the MIDlet (§2.1). The ABML rules we outline below are contained in our *messenger* policy, from which a Java `Policy` is computed that generates an execution monitor. Both the MIDlet and its monitor execute on the device as a separate threads inside the Polymer framework, so the device's application permissions for Polymer (for *connections* and *user data*) must all be set to "allow".

[1] **Bluetooth backdoor attack:** MIDlets can transmit data to and from the device via its Bluetooth serial port. They must be signed to establish a Bluetooth connection, but can use an open serial port without being signed. Sensitive data (e.g., e-mail, SMS messages, Personal Information Manager (PIM) contacts) can be captured and transmitted to other in-range devices, some of which may be attacker-controlled. To prevent this, the following rule should be applied: *"the target may send data to the Bluetooth serial port only if it has established a client-side Bluetooth connection with a paired device"*.

Policy 1. Policy to prevent a MIDlet using a Bluetooth backdoor

ABML rule	Connection b, c: device.data.Send(Stream s, c)|c.address startsWith "btspp://" → device.service.Connect(b)|b.address := c.address:
Rule format	var_1: var_2: $\mathsf{H}\{\mathsf{E}_2(\mathsf{O}_2)\|\mathsf{cond}_{(2)}\} \rightarrow \mathsf{E}_1(\mathsf{O}_1)\|\mathsf{cond}_{(1,2)}$:
Triggering event	device.data.Send(Stream s, Connnection c)

Monitored calls	`javax.microedition.io.Connector.<init>`
	`net.rim.device.api.bluetooth.BluetoothSerialPort.<init>`

A significant level of user interaction is required to establish such a connection, so users are unlikely to be coerced into doing so easily. The abstract event localdevice.data.Send(Stream s, Connnection c) triggers Policy 1. It relates to the low-level actions a MIDlet may use to stream data to some connection and is conditional on that connection being to a Bluetooth serial port. If the policy's conclusion evaluates to `true`, the monitor has recorded that a legal client-side Bluetooth connection has been established with a paired device and the Send event's address parameter matches that of the established connection – invocation is allowed. Otherwise, Send is suppressed and the MIDlet's execution continues without that data being transmitted.

[2] **SMS interception attack:** Sending and receiving an SMS message uses the MIDP 2.0 standard [12] and so does not require an application to be signed. Once the BlackBerry user has agreed to its standard prompt *"Allow Network Access?"*, they do not receive further warnings, even for subsequent executions. An untrusted MIDlet can create an 'SMS channel', which remains in place where the attacker has programmed it to run as a background process on receiving an *exit event*. We therefore add the rule: *"the target may send an SMS message only if the data that message contains was entered manually by the user and the user invoked the sending of that SMS by pressing the device's trackwheel"*.

If the application attempts to send an SMS message containing data that was not user-entered, invocation of internet.data.Send(SMSMessage s) is suppressed. Policy 2 uses the construct $\mathsf{H} \rightarrow \mathsf{R}$, where R refers to an abstract event sequence that must have already occurred (and any conditions on its occurrence satisfied) for H to be invoked. If this is not the case, *any* event which relates to sending an SMS message to the internet domain is prevented.

Our extensions to the Polymer engine determine the context of the triggering event by monitoring the MIDlet getting data from a text field after the user has

Policy 2. Policy to defend the BlackBerry against SMS interception

ABML rule	SMSMessage s: Text r, t: internet.data.Send(s) $\rightarrow$ device.keypad.Press(r);; device.trackwheel.Click(); device.gui.GetText(t)|t := s.body $\wedge$ t contains r:
Rule format	var_1: var_2: var_3: $H\{E_4(O_4)\} \rightarrow E_1(O_1);;\ E_2(..);\ E_3(O_3)\|\text{cond}_{(3,1,2)}$:
Triggering event	internet.data.Send(SMSMessage s)

Monitored calls	`net.rim.blackberry.api.sms.OutboundMessageListener.notifyOutgoingMsg(Msg m)`
	`net.rim.device.api.system.EventInjector.KeypadEvent(..)`
	`net.rim.device.api.system.EventInjector.TrackwheelEvent(THUMB_CLICK, .., ..)`
	`net.rim.device.api.ui.component.TextField.getText(..)`

manually entered it and pressed an interface button to send an SMS message containing that text. Our compiler creates empty objects (O of type `Text`) that refer to all *live instances* of `Text` pertaining to the user's interaction. Further details on this process are provided in Section 4.2.

This policy uses a *strong* temporal order between E_2 and E_3. If events which this sequence does not reason about occur in between these, the policy is violated and its triggering event supressed. The method call `TextField.getText(..)` is tied as closely as possible to a `TrackwheelEvent` that uses a 'thumb click'. More complex reasoning here would require the policy to model the MIDlet's user interface, which is not possible for *unseen malware*.

[3] **HTTP proxy attack:** Unsigned MIDlets can create TCP connections on the BlackBerry, again prompting the user with the BlackBerry's *"Allow Network Access?"* dialog only once. Attack code can then use the device as a proxy for traffic; the attacker often having the intention of accessing illicit material or performing denial of service attacks.

In order to mitigate this, we construct a policy to analyse the messenger's use of the device's Internet connection. This states: *"the target may send an HTTP response* (r) *to the network only if it has not previously received an HTTP request* (n) *whose host's name matches that of* r*'s host; where* r *was proceeded by the sending of an HTTP request* (p) *and the receipt of an HTTP response* (q) *whose hosts match"*.

Policy 3. Policy to prevent a MIDlet using the BlackBerry as an HTTP proxy

ABML rule	HttpResponse r, q: HttpRequest n, p: internet.data.Send(r) $\rightarrow$ internet.data.Receive(n)|n.host $\neg =$ r.host;; internet.data.Send(p);; internet.data.Receive(q)|p.host := q.host:
Rule format	var_1: var_2: var_3: var_4: $H\{E_4(O_4)\} \rightarrow E_1(O_1)\|\text{cond}_{(3,1)};;\ E_2(O_2);;\ E_3(O_3)\|\text{cond}_{(4,2)}$:
Triggering event	internet.data.Send(HttpResponse r)

Monitored calls	`javax.microedition.io.HttpConnection.openInputStream(..)`
	`javax.microedition.io.HttpConnection.openDataInputStream(..)`
	`javax.microedition.io.OutputStream.openOutputStream(..)`
	`javax.microedition.io.DataOutputStream.openDataOutputStream(..)`

The monitor constructed here analyses request calls to `HttpConnection` and ascertains whether a related HTTP response is ever sent to the network. Our enforcement model constructs `HttpRequest` and `HttpResponse` objects whenever abstract events whose arguments match these are triggered (§4.2). These encapsulate any data being sent or received over HTTP by the messenger application.

4 Policy Compilation and Enforcement

Our compiler translates ABML policies into Java classes (of type `Policy`). Translation from an ABML policy to a `Policy` class is performed in a compositional manner, so incremental changes can be handled with ease. Users may wish to weaken a policy where it denies some program behaviour they wish to permit, or strengthen it where they learn of some vulnerability in the target MIDlet.

4.1 Synthesising Monitors from ABML Specifications

Each `Policy` output by our compiler contains a method called `query()`, in which every rule is implemented by enumerating `case`s in a `switch()` statement. A policy is triggered where an event's signature is matched by precisely one `case`. The method then returns a `Suggestion` object which indicates how that event should be dealt with. At the highest level, our compiler's algorithms operate as follows to compose a `Policy`.

1. $\forall$ E, select the pre-compiled `AbstractAction` class which E refers to. Insert appropriate `import` statements into `Policy` to provide access to these classes.
   (a) $\forall$ O in E, create an empty class of type `AbstractData` with instance variables to reflect O's parameters. Insert appropriate `import` statements.
      i. Provide appropriate accessor and mutator methods to access O's parameters.
2. Populate `query()`'s `switch()` statement with one `case` per E. Set that `case`'s trigger to the `AbstractAction` selected in (1).
   (a) If E is non-atomic, prepend each `case`'s trigger with the keyword "`abs`"[4].
3. $\forall$ cond in E, declare one boolean variable, $\mathtt{b}_n$, setting its value to `false`.
4. $\forall$ E, such that E's arguments contain O, insert a reference to that O inside that `case`'s trigger[5].
5. $\forall$ E, construct one conditional statement for each *related* $\mathtt{b}_n$ in that E's `case`. Populate it with an expression to represent E's cond[6].
   (a) $\forall$ E such that cond evaluates to `true`, set *related* $\mathtt{b}_n$ = `true`.
6. Compute reasoning constructs to recover from a rule violation. $\forall$ rule:
   (a) $\forall$ cond in a rule's *premise*:
      i. If cond = `true`, return an `OKSug` (e.g., that event can be executed);
      ii. Else if cond = `false`, this event is irrelevant: return an `IrrSug`.

[4] E.g., internet.data.Send(..) → `case <abs void internet.data.Send(..)>:`
[5] E.g., internet.data.Send(Request r) → `case <abs void internet.data.Send(r)>:`
[6] E.g., `head.startsWith(''http://'')`

(b) ∀ cond in a rule's *conclusion*:
   i. If R = ⊥, return `Suggestion` to halt the MIDlet (`HaltSug`);
   ii. Else if R = cond, and cond is `false`, halt the MIDlet;
   iii. Else if R = E:
      – If b_n = `false`, this event is irrelevant: return an `IrrSug`;
      – Else return `ExnSug` to suppress E but continue execution.

Rule precedence is determined by the order rules are listed in a policy. A monitor queries its first rule implementation and always follows the `Suggestion` of that `rule` if it considers the trigger action to be security-relevant (e.g., an `ExnSug` or a `HaltSug` is returned, but not an `IrrSug` or an `OKSug`). Otherwise, the `Suggestion` of the second `rule` is followed, and so on. Authors whom edit policies in a textual manner should therefore take care, though we presume most will modify policies using our specification GUI.

4.2 ABML Policy Enforcement

Figure 2 illustrates our monitoring model and depicts a target application and an execution monitor in execution on the JVM. The three grey-shaded elements represent the classical execution monitoring process, as was first described by Schneider [16], and as is implemented by Polymer [3,2]. All events invoked by the target program are analysed by the execution monitor: events which the policy does not reason about are executed without analysis and those which it reasons about are apprehended if any conditions on them evaluate to `false`.

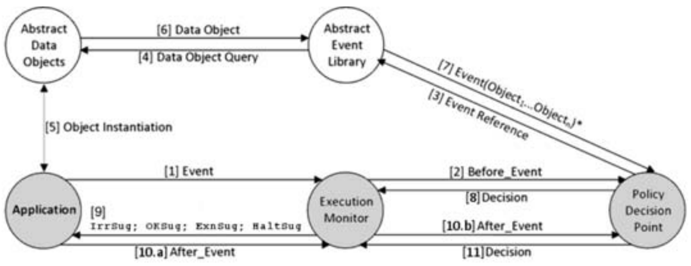


Fig. 2. Operation of our ABML policy enforcement mechanism

Our extensions to this approach (white-shaded elements) "investigate" the context in which an event occurs to achieve more precise separation between *legitimate* and *misuse* behaviours. This process works as follows:

– Empty objects, derived from any O in a rule at compile-time, are created. They extend `AbstractData` and contain the necessary instance variables to represent O. This mapping is pre-defined in an XML file.
– Accessors and mutators for each O are set by our compiler and are mapped to any ABML event (E) which refers to it.

- Each `AbstractData` object is instantiated when a MIDlet performs a related event which a `Policy` reasons about (cf. step 5 in Figure 2):
  - A new class is created, whose unique reference identifies the data used by that call;
  - Garbage-collected instances of this class are checked for in `classMap`, which is a hash map that collects class names and their references to other classes;
  - If an object has not been garbage-collected then it is a live instance. Its reference is set to its associated instance variable in `Policy`.
- A `Policy` is able to query that object using the accessors and mutators it provides.

Preventing calls which bypass APIs. Malicious programs may try to load native code in order to mirror the functions of an API and so effectively bypass it. Our approach can avoid such an attack. MIDlet bytecode is "pre-verified" by the CLDC Class Verifier, which acts to sandbox an application by inserting certain security checks as class file annotations to be carried out when the device loads the class. Changes to bytecode after this has occurred are detected at runtime by the JVM, which will reject the class [6]. Furthermore, our system sets a Java `SecurityManager` policy which itself does not set a `checkLink` permission for any native library. Attempts at executing the `load()` or `loadLibrary()` methods therefore result in a `SecurityException`.

4.3 Performance Analysis

Expressing policies: We have tested our language and enforcement engine using the attacks presented (§3.2). In order to mitigate these, our compiler generated 356 lines policy code, split between three `Policy` classes and three `Combinator` classes. The latter enforce rule precedence. An equivalent ABML requires only 15 lines of code: 5 lines of global variable declarations and 10 lines of ABML rule specifications (an arithmetic mean of 5 lines per attack). The policies ABML can express are more readable than their imperative counterparts.

Mitigating information flow: Our model enables a class of policies which prevent malicious programs from leaking sensitive information. We ensure that if an information flow constraint cannot be mitigated, the events which cause it are denied. Execution monitoring cannot capture all cases of information flow [16]; full information flow analysis is required to achieve this [13]. The significant difference between our language and others when considering information flow is its ability to specify live instances of data as arguments to events.

Execution-time overhead: As mobile devices have limited computing power, we examine the costs of enforcing policies using our approach at:

- **Download-time:** Scanning a MIDlet's bytecode for API calls is dependant on the number of calls it makes. Our test code contained 83 such calls, which our scanner took 4.4s to extract and add to an event definition file.

– **Specification-time:** Compilation of a `policy` is also dependant on its complexity. Compiling our rules to prevent attacks 1 through 3 (§3.2) took 1.2s.
– **Load-time:** The total time to instrument every method in the Java ME and BlackBerry APIs (i.e., the 7352 methods in the 933 classes in the `javax` and `net.rim` packages of the BlackBerry API v.4.5.0) was 185s, or an average 25ms per instrumented method. This cost is reasonable because library instrumentation need only be performed once per download.
– **Runtime:** The cost of transferring control to and from a policy while executing a target is very low (approximately 1.44 ms per policy decision point). Therein, the run-time overhead of monitoring is is dependent on the complexity of the security policy.

5 Related Work

The foundations of execution monitoring are described by Schneider in [16], where a formal treatment is given to techniques which analyse the actions of a target program and terminate it where it violates a policy. Ligatti [1,4] have extended this model to suppress program actions, allowing program execution to continue after a policy is violated. This class of monitor is modelled by the *edit automaton*, which is implemented by Polymer [2,3]. Its policies are separated from the target program and are therefore easy to maintain, re-use, or compose into a hierarchy. Furthermore, it allows only *sound* execution monitors to be computed. A weakness of Polymer is its input language (a constrained Java syntax), which has high expressive power but is difficult for end-users to write policies in.

Much work has been undertaken in developing languages for specifying process behaviours that signify an intrusion. Sekar et al. devised Behaviour Modelling Specification Language (BMSL) [17,18], a more user-friendly policy language for reasoning about a program's execution through combining system call abstractions. BMSL can express policies that capture the values of arguments to events, but a policy author must forecast the strong temporal order of trigger and recovery events: an infeasible expectation of most.

Later work on policy enforcement seems not to have focussed much on enhancing its precision. Castrucci et al. [5] describe how to monitor MIDlets, although their policy language requires imperative reasoning using atomic method call signatures and cannot precisely capture the temporal order or the context of event invocations. Those which have attempted this [9] have done so in a low-level manner, focussing on system calls on traditional hosts. It is often difficult, or even impossible, to attribute a system call sequence to a particular program event [15].

6 Conclusion

We have designed a user-operable sandboxing technique to control the execution of untrusted mobile software. It consists of an abstract policy language and an enforcement model based on Polymer. Our language is more expressive than other policy languages, but uses purely declarative constructs. Its policies require lower development effort and can be composed and applied incrementally.

ABML's main contribution is its association of atomic events to program data and its mapping of this process to suitable abstractions, which helps to increase accuracy and reduce vulnerabilities and redundancies. We have provided an enforcement model that is more powerful than existing execution monitors – it is aware of the context in which an event has occurred and so broadens the functionalities of the target application. Our future work will prove that the policies we compute are at least as strong as the ABML policies which specified them when enforced by our model and that their translation into a `Policy` class is sound. This will enable users to place guarantees in our work.

References

1. Bauer, L., Ligatti, J., Walker, D.: More enforceable security policies. In: FLoC 2002: Proceedings of the 2002 Workshop on Foundations of Computer Security, pp. 95–104 (2002)
2. Bauer, L., Ligatti, J., Walker, D.: A language and system for composing security policies. Technical Report TR-699-04, Princeton University (2004)
3. Bauer, L., Ligatti, J., Walker, D.: Composing security policies with Polymer. In: PLDI 2005: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation, pp. 305–314. ACM Press, New York (2005)
4. Bauer, L., Ligatti, J., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policies. International Journal of Information Security 4(1-2), 2–16 (2005)
5. Castrucci, A., Martinelli, F., Mori, P., Roperti, F.: Enhancing Java ME Security Support with Resource Usage Monitoring. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 256–266. Springer, Heidelberg (2008)
6. Java Community Process: CLDC 1.1 Specification, `http://jcp.org/aboutJava/communityprocess/final/jsr139/`
7. Erlingsson, U., Schneider, F.B.: IRM enforcement of Java stack inspection. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 246–259. IEEE Computer Society Press, Washington (2000)
8. Evans, D., Twyman, A.: Flexible policy-directed code safety. In: SP 1999: Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp. 32–45. IEEE Computer Society Press, Washington (1999)
9. Giffin, K., Jha, S., Miller, B.: Efficient context-sensitive intrusion detection. In: NDSS 2004: Proceedings of the 11th Annual Network and Distributed Systems Security Symposium. Internet Society Press, Reston (2004)
10. Research in Motion: Blackberry Simulators, `http://na.blackberry.com/eng/developers/downloads/simulators.jsp`
11. Sun Microsystems: Connected Limited Device Configuration (CLDC), `http://java.sun.com/products/cldc/`
12. Sun Microsystems: Mobile Information Device Profile (MIDP), `http://java.sun.com/products/midp/`
13. Myers, A.C., Liskov, B.: Protecting privacy using the decentralized label model. ACM Transactions on Software Engineering Methodology 9(4), 410–442 (2000)
14. Forum Nokia: Java Security Domains, `http://wiki.forum.nokia.com/index.php/JavaSecurityDomains/`

15. Provos, N.: Improving host security with system call policies. In: Proceedings of 12th USENIX Security Symposium, pp. 128–146. USENIX Press, Washington (2003)
16. Schneider, F.B.: Enforceable security policies. ACM Transactions on Information Systems Security 3(1), 30–50 (2000)
17. Sekar, R., Uppuluri, P.: Synthesizing fast intrusion prevention/detection systems from high-level specifications. In: SSYM 1999: Proceedings of the 8th USENIX Security Symposium. USENIX Association, Berkeley (1999)
18. Sekar, R., Venkatakrishnan, V.N., Ram, P.: Empowering mobile code using expressive security policies. In: NSPW 2002: Proceedings of the 10th New Security Paradigms Workshop, pp. 61–68. ACM Press, New York (2002)

Extending the Belgian eID Technology with Mobile Security Functionality

Jorn Lapon[1], Bram Verdegem[1], Pieter Verhaeghe[2], Vincent Naessens[1], and Bart De Decker[2]

[1] Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Gent, Belgium
[2] Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium

Abstract. The Belgian Electronic Identity Card was introduced in 2002. The card enables Belgian citizens to prove their identity digitally and to sign electronic documents. Today, only a limited number of citizens really use the card in electronic applications. A major reason is the lack of killer functionality and killer applications.

This paper presents two reusable extensions to the Belgian eID technology that opens up new opportunities for application developers. First, a secure and ubiquitously accessible remote storage service is presented. Second, we show how the eID card can be used to issue new certificates. To demonstrate the applicability and feasibility of both extensions, they are combined in the development of a secure e-mail application. The proposed solution offers strong privacy, security and key management properties while increasing the accessibility of confidential e-mail compared to existing solutions (such as PGP and S/MIME).

Keywords: Identity Technology, Security, Privacy, Mobile Access.

1 Introduction

In 2002, Belgium has introduced an electronic identity card (eID) [1] as one of the first countries in Europe. The card enables individuals to prove their identity digitally and to sign electronic documents. The Belgian eID card opens up new opportunities for the government, their citizens, service providers and application developers. Although many eID applications have been developed, the success of the Belgian eID technology is still limited. A major reason is the lack of essential functionality and, more importantly, real killer applications. Moreover, the use of the current eID card involves a few security and privacy hazards [2,3].

This paper presents two enhancements to the current Belgian eID technology while addressing certain privacy shortcomings. A first extension defines a service that allows users to store and update sensitive data (such as passwords, keys, tickets, . . .) securely at a remote location. The service is ubiquitously accessible with the Belgian eID card. Moreover, the data that is kept at the server is useless to internal and external attackers. A second extension defines the creation and use of proxy certificates that are certified by means of the Belgian eID card. Hence, an individual can create new certificates that can be used to send confidential messages, to setup a mutually authenticated secure channel between two individuals, etc. An external certificate authority is no longer required to issue these certificates.

Both extensions are bootstrapped by means of the Belgian eID Card and can be integrated in many applications. To demonstrate their usefulness, both extensions have been incorporated in a ubiquitously accessible secure e-mail service. This service allows individuals to send confidential (encrypted) and signed messages using certificates that can be validated by certificate chains. The approach is more secure than PGP (which is based on trust levels) and tackles some key management problems in S/MIME (i.e. confidential e-mail can be retrieved from any location at which a web browser, Internet access and a card reader is available).

The paper is structured as follows. Section 2 gives an overview of the Belgian eID card technology. Section 3 describes the notation used in the rest of the paper. Two extensions to the Belgian eID technology are proposed in section 4 and section 5. Both extensions are evaluated in section 6 and validated through the development of a secure e-mail application in section 7. Finally, the paper draws some conclusions and describes directions for future research.

2 Belgian Electronic Identity Card Technology

The *Belgian eID card* is a smart card that allows Belgian citizens to prove their identity visually and digitally and to sign electronic documents [4]. The eID card contains three files: (1) a digital picture of the citizen, (2) an identity file which contains the basic identity information and a hash value of the picture file; this file is signed by the National Registry, (3) an address file which contains the citizen's current residence; it is signed by the National Registry together with the identity file to guarantee the link between both files.

Two private keys SK_{Auth} and SK_{Sig} are stored in the eID card. These keys are used for digital authentication and signing respectively. They are stored in a tamper-proof part of the chip and can be activated with a PIN code. Each corresponding public key (PK_{Auth} and PK_{Sig}) is certified by a certificate. Each certificate also keeps the name of the card holder and his nation-wide identification number (i.e. the National Registry Number or *NRN*).

The Belgian government offers a *middleware* package [5,6] to facilitate interaction with the eID card. The middleware contains a GUI to enable end-users to read the files and the certificates that are stored in the eID card and to change the PIN code. Moreover, the middleware acts as an intermediary for all accesses to the eID card by other applications. If a document has to be signed, the middleware passes a hash of the document to the card. Similarly, a hash of the challenge is passed to the card for authentication purposes. When an application wants to authenticate or sign a document with the eID card, the middleware asks the user for a PIN code and forwards it to the eID card. The middleware can also check the validity of certificates (using CRL or OCSP). Note that the middleware is not essential: an application can also implement the middleware functionality and directly interact with the card.

The certificates on the eID card are part of a larger hierarchical infrastructure, the *Belgian Public Key Infrastructure* [7]. The signature and authentication certificates are issued by a *Citizen_CA* and the certificates of each *Citizen_CA* are issued by the *Belgium_Root_CA*, that is found at the top of the eID hierarchy. In addition to the CA hierarchy, the PKI defines Certificate Revocation Lists [8] that contain the serial numbers of revoked certificates issued by that CA.

The government aims at encouraging the use of the eID card in both *e-government and commercial applications.* Currently, most applications use the Belgian eID card for setting up an SSL connection with mutual authentication. Many other applications (such as physical access control) only retrieve the identity information stored on the card.

3 Notations

The following notation is used throughout this paper:

- $\mathrm{P}_1 \leftrightarrow \mathrm{P}_2 : NRN \leftarrow \mathsf{authenticate}_{\mathsf{eID}}()$ represents an interactive protocol, in which P_1 uses SK_{Auth} in the eID card to authenticate to P_2. As a result, P_2 obtains P_1's NRN.
- $\mathrm{P} : sig \leftarrow \mathsf{sign}_{\mathsf{eID}}(\mathsf{hash}(M))$ denotes a user signing the hash of message M, with SK_{Sig} in the eID card.
- $\mathrm{P} : K \leftarrow \mathsf{createSymmetricKey}(PRG, seed)$ denotes the generation of a secret/symmetric key, based on a pseudorandom number generator PRG and a *seed*. Use of the same PRG and *seed* will always generate the same key.
- $\mathrm{P} : \mathsf{store}(data; index)$ means that *data* is stored in a database at location *index*.

4 Mobile Access to Secrets

A major challenge in today's society is to enable access to sensitive data (e.g. personal information, secret keys, ...) from various locations. Data must be stored securely on an easily accessible remote server. Data is typically encrypted by the owner before it is stored on the server. Hence, the encrypted data is useless to the server or any adversary that may have access to it. In this section a scheme is discussed based on the Belgian eID card for securely storing sensitive data. A trivial solution consists of encrypting the data with the public key of the authentication certificate. Whenever needed, the ciphertext is fetched from the server and decrypted using the corresponding private key.

However, encryption/decryption with the Belgian eID card is not possible. Moreover, when the card is lost or renewed, decryption of previously encrypted information is no longer possible. Therefore, a new mechanism is proposed that uses the Belgian eID card as a bootstrap. A secret/symmetric key is derived from

Table 1. Storing sensitive data remotely

$\mathsf{storeSensitiveData}(data,\ tag)$:

(1)	U	:	$H_{keyGen} \leftarrow \mathsf{hash}_{\mathsf{A}}('KEYGEN' \| NRN \| CardNumber \| otherUserInfo)$
(2)	$\mathrm{U} \leftrightarrow \mathrm{C}$	:	$sig \leftarrow \mathsf{sign}_{\mathsf{eID}}(H_{keyGen})$
(3)	U	:	$K_S \leftarrow \mathsf{createSymmetricKey}(PRG, \mathsf{hash}_{\mathsf{B}}(sig))$
(4)	U	:	$E_{tag} \leftarrow \mathsf{encrypt}(tag, K_S)$
(5)	U	:	$R \leftarrow \mathsf{generateRandom}()$
(6)	U	:	$K_D \leftarrow \mathsf{createSymmetricKey}(PRG, \mathsf{hash}_{\mathsf{C}}(sig \| R))$
(7)	U	:	$E_{data} \leftarrow \mathsf{encrypt}(data, K_D)$
(8)	$\mathrm{U} \leftrightarrow \mathrm{S}$	:	$NRN \leftarrow \mathsf{authenticate}_{\mathsf{eID}}()$
(9)	$\mathrm{U} \rightarrow \mathrm{S}$	:	(E_{data}, R, E_{tag})
(10)	S	:	$index \leftarrow \mathsf{hash}_{\mathsf{D}}(NRN \| E_{tag})$
(11)	S	:	$\mathsf{store}([E_{data}, R]; index)$
(12)	$\mathrm{U} \leftarrow \mathrm{S}$	:	$(Timestamp, \mathsf{sign}_{\mathsf{S}}(\mathsf{hash}_{\mathsf{E}}([Timestamp, E_{tag}, E_{data}, R])))$

a signature generated by the eID card. This secret key is used for encrypting data before it is stored. When a user wants to retrieve his confidential information, the encrypted information is fetched from the server and decrypted with the secret key, regenerated using the same eID card. In the following paragraphs, the protocols for storing and retrieving sensitive data are discussed in more detail.

Storing sensitive data. Table 1 shows the steps for storing sensitive data. The user provides the *data* to be stored and a secret *tag* that serves as a name or alias of the data.

First, the Belgian eID card is used to generate secret keys as follows. A message with a fixed format is hashed using the middleware (1). The message consists of a header *'KEYGEN'*, the *NRN*, the serial number of the eID card and possibly some other user information, making the message card specific. The hash, H_{keyGen}, is then signed with the signature key on the eID card resulting in a 1024 bits signature *sig* (2). The fixed format prevents that an adversary obtains *sig* by requesting a signature on a forged message; any message to be signed by the eID card should not match this format, except in this protocol. Subsequently, *sig* is used as the seed for generating two new symmetric keys, namely K_S and K_D. To ensure that the same keys are generated, independent of the platform, a specific pseudo-random generator *PRG* is passed as a parameter of the createSymmetricKey function; also, the hash-functions used should be fixed. K_S is used for encrypting the secret *tag* associated with the data. The key is derived from the hash of *sig* (3-4). The encrypted tag E_{tag} is used in step 10 of the protocol to derive an *index* to a record in the server database. K_D is used for encrypting the *data*. Each time information is stored (i.e. each time the protocol is invoked), a new random number R is associated with it (5). From the hash of R and *sig* the encryption key K_D is derived (6) with which the *data* is encrypted into the ciphertext E_{data} (7). In each execution of storeSensitiveData the data will be encrypted with another key; this prevents linking of the same encrypted data that is stored in more than one location.

Next, the encrypted data E_{data} is stored on a remote server S. First, the user authenticates with his eID, to ensure that later only the owner U can retrieve his data (8). Next, U sends the encrypted tag E_{tag} and the encrypted data E_{data} to S (9). The *NRN*, disclosed during the authentication, is hashed together with E_{tag} and will be used as an *index* to the information that is stored in the server database (10). The use of the encrypted secret tag, E_{tag}, ensures the user's privacy, while making the *index* tag specific. A different *tag** will result in another index. If the server S is trustworthy and does not store the user's *NRN* nor the E_{tag}, dictionary attacks on the *index* are no longer feasible. An adversary with full access to the server data cannot link any data to a particular citizen.

Finally, the database stores a record with the encrypted data and the random number R at location *index* (11). Although useless to the server or any other adversary, the random number R is necessary for the owner of the data to derive the correct symmetric key for decrypting E_{data}. A receipt is sent to the user, certifying the proper storage of the encrypted data (12).

Table 2. Retrieving sensitive data remotely

retrieveSensitiveData(tag):

(1) U	:	$H_{keyGen} \leftarrow \mathsf{hash_A}('KEYGEN' \| NRN \| CardNumber \| otherUserInfo)$
(2) U ↔ C	:	$sig \leftarrow \mathsf{sign_{eID}}(H_{keyGen})$
(3) U	:	$K_S \leftarrow \mathsf{createSymmetricKey}(PRG, \mathsf{hash_B}(sig))$
(4) U	:	$E_{tag} \leftarrow \mathsf{encrypt}(tag, K_S)$
(5) U ↔ S	:	$NRN \leftarrow \mathsf{authenticate_{eID}}()$
(6) U → S	:	$\mathsf{requestRecord}(E_{tag})$
(7) U ← S	:	$record \leftarrow \mathsf{getRecord}(\mathsf{hash_D}(NRN \| E_{tag}))$
(8) U	:	$K_D \leftarrow \mathsf{createSymmetricKey}(PRG, \mathsf{hash_C}(sig \| record.R))$
(9) U	:	$data \leftarrow \mathsf{decrypt}(record.E_{data}, K_D)$

Retrieving sensitive data. When the data has been stored on the remote server, it can be retrieved by the owner from anywhere (see table 2). First, the hash H_{keyGen} is regenerated (1) and signed with the eID card (2). With the signature *sig*, the secret key K_S (3) is regenerated for encrypting the secret tag (4). Next, the user authenticates with his eID card (5). The encrypted secret tag, E_{tag} is sent to the server S and the corresponding record requested (6). The server S fetches the record with $index = \mathsf{hash_D}(NRN||E_{tag})$ from his database; the record, comprising of encrypted data and the random number, is sent to U (7). U can now regenerate the data specific secret key K_D from the hash of *sig* and R (8). Finally, U decrypts E_{data} with the secret key K_D (9).

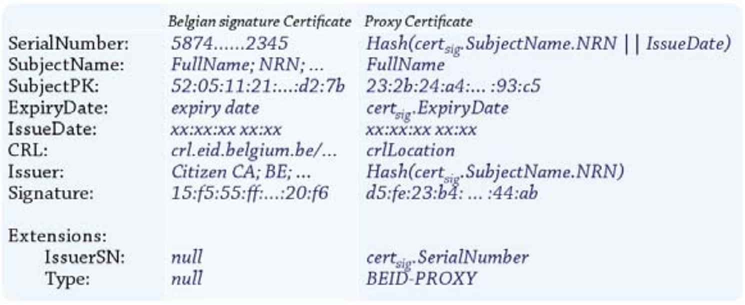
Recovery of keys. This scheme exploits the property that a deterministic signature algorithm is implemented in the eID card. When creating a signature with the eID card, the same input (H_{keyGen}), always results in the same signature *sig*. As such, using the same *PRG* and the same hash-functions ($\mathsf{hash_A}$ .. $\mathsf{hash_D}$), the same secret keys K_S and K_D (for a certain R) are always regenerated. However, in case of loss or renewal of the eID card, PK_{Sig}, SK_{Sig} and *CardNumber* will have changed and the generated signature *sig** no longer matches the signature *sig* (generated by the previous eID card), impeding the localization and decryption of the stored information. Therefore, to protect important data, a secured backup of the signature *sig* is created the first time this signature is generated. In case of loss or renewal, *sig* is restored from the backup and the keys can be recovered. This secured backup could be provided by a key escrow service.

5 Proxying the Belgian eID

As illustrated above, it is now possible to perform symmetric encryption based on the Belgian eID card. However, when other parties are involved, asymmetric encryption may be required. Therefore, a second encryption scheme based on the eID card is proposed.

Proxy certificates. Although the eID card itself cannot encrypt nor decrypt data, the right to do this can be delegated to the host. This restricted delegation and proxying to another entity is achieved through proxy certificates [9]. A proxy certificate is an extended version of a normal X.509 certificate. Proxy certificates can be derived from and signed by a normal X.509 certificate or by another proxy certificate. Once a proxy certificate is created, the proxy certificate and its corresponding private key can be used for asymmetric encryption resp. decryption.

Modified standard. The standards for proxy certificates, as defined in the RFC 3820, impose some problems. Therefore, some modifications have to be made (see figure 1). First, according to the RFC, proxy certificates should be issued by the authentication certificate, since only this certificate of the Belgian eID contains the required *key-usage* attribute. The *key-usage* attribute defines the purpose of the public key contained in the certificate. However, using the authentication key SK_{Auth} is less secure than using the signature key SK_{Sig} since the PIN is only required for the first authentication (Single Sign On feature), while it is required for every signature. Therefore, SK_{Sig} is used to issue proxy certificates. Second, the Belgian legislation prohibits to store the *NRN*. However, the *NRN* is stored in the subject field of the eID certificates. This implies that the eID certificates may not be stored. However, the proxy certificate standard defines that the subject of the issuing certificate is copied into the issuer and subject fields of new proxy certificates. To solve this problem, the name of the owner is copied into the subject field and the hash of the *NRN* is copied into the issuer field instead of the subject of the eID certificate. For validation purposes, the serial number of the issuer certificate is also included in the certificate (i.e. *issuerSN*). Additionally, another extra attribute indicates the type: `BEID-PROXY`. In this scheme, we assume that when the eID certificate is revoked (e.g. in case of loss or theft), the issued proxy certificates are no longer valid. Moreover, the proxy certificate must expire before the expiration date of the eID certificate. The protocol in table 3 demonstrates the creation of a BeID proxy certificate. The user U generates an asymmetric key-pair (1). A serial number is generated from the hash of the *NRN* and the *issueDate* of the new proxy certificate (2-3).



	Belgian signature Certificate	Proxy Certificate
SerialNumber:	5874......2345	Hash(cert$_{sig}$.SubjectName.NRN || IssueDate)
SubjectName:	FullName; NRN; ...	FullName
SubjectPK:	52:05:11:21:...:d2:7b	23:2b:24:a4:... :93:c5
ExpiryDate:	expiry date	cert$_{sig}$.ExpiryDate
IssueDate:	xx:xx:xx xx:xx	xx:xx:xx xx:xx
CRL:	crl.eid.belgium.be/...	crlLocation
Issuer:	Citizen CA; BE; ...	Hash(cert$_{sig}$.SubjectName.NRN)
Signature:	15:f5:55:ff:...:20:f6	d5:fe:23:b4: ... :44:ab
Extensions:		
IssuerSN:	null	cert$_{sig}$.SerialNumber
Type:	null	BEID-PROXY

Fig. 1. Content of the modified proxy certificate

Table 3. Create a new proxy certificate

createBeIDProxy$([attributes])$:

(1) U	:	$(SK_U, PK_U) \leftarrow$ generateKeyPair()
(2) U	:	$issueDate \leftarrow$ getDate()
(3) U	:	$serialNb \leftarrow$ hash$(NRN\|\|issueDate)$
(4) U	:	$proxyCert \leftarrow$ generateProxy(BEID-PROXY, $cert_{Sig}, serialNb,$ $PK_U, issueDate, crlLocation, [attributes]; SK_{Sig}$)

The *NRN* in the hash avoids collisions, while the *issueDate* enables users to have more than one proxy certificate. A proxy certificate *proxyCert* is then generated and certified with SK_{Sig} of the eID card (4).

Revocation. A special purpose server R_p can publish revoked proxy CRLs. To revoke a proxy certificate (cfr. table 4), the user authenticates with his eID card (1) and sends the proxy certificate he wants to revoke (2). If the issuer corresponds to the eID signing certificate (i.e. hash$(cert_{Sig}.NRN)$ = *proxyCert.Issuer*), the proxy certificate is revoked by adding its serial number to the latest CRL.

Table 4. Revoke a proxy certificate account

revokeBeIDProxy$(proxyCert)$:

(1) U $\leftrightarrow$ R$_p$	:	$NRN \leftarrow$ authenticate$_{\mathsf{eID}}$()
(2) U $\rightarrow$ R$_p$	:	revokeCertificate$(proxyCert.serialNumber)$
(3) S	:	if (hash$(cert_{Sig}.SubjectName.NRN) \neq proxyCert.Issuer$) abort
(4) U $\leftarrow$ S	:	$true \leftarrow$ addToCRL$(proxyCert.serial)$

Validation. A receiver validates a new proxy certificate by checking the validity period, its revocation status and by verifying the rest of the certificate chain. Since the hash of *NRN*is kept in the issuer field, name chaining (cfr. RFC 3280 [10]) for certification path validation will fail. However, the extra attribute *issuerSN* included in the certificate binds the eID certificate to the proxy certificate. The first step in creating the certification path is thus modified. The serial number of the eID certificate must match the *issuerSN* in the proxy certificate. To comply with Belgian legislation, the eID certificate is removed after validation. Hence, future validation is not possible. However, verifying the validity period and the revocation status suffices. This can be performed as the proxy certificate is stored at a trusted location. Additionally, the revocation status of the eID certificate can be verified by checking the *issuerSN* of the proxy certificate in the CRLs of the Belgian eID.

Belgian citizens can now create legitimate *proxy certificates* themselves, that can be used in many applications. Moreover, once a proxy certificate has been created, the Belgian eID is no longer required.

6 Discussion

The extensions discussed above promise new opportunities for the Belgian eID technology. Not only can the eID card be used for digital signatures or authentication, the eID technology can also be used to store and to retrieve sensitive data. The user only needs his card to access the encrypted data that is stored on the remote server. However, adversaries might try to retrieve the secret keys by continuously sending challenges to the eID card. Our solution tackles the thread by using the signature key in the eID card. The latter requires a PIN for every signature in contrast to the authentication key, which only requires a PIN once. Moreover, the fixed format of the message allows to detect trojan horses or malicious applications that request users to sign a certain message.

To support recovery of sensitive data if the eID card is lost or invalid, a secure backup of the signature *sig* needs to be created the first time this signature is generated. However, the user remains responsible for making the secure backup. An alternative approach is to support a key-escrow mechanism [11,12]. The secret *sig* is then split into n parts using a secret sharing algorithm and each part is stored on a different escrow server. To reconstruct the secret, all n parts are retrieved from the escrow servers and the interpolation of the parts results in the secret. To ensure that users only obtain their own keys, eID authentication can be used with the escrow servers. Additionally, a hash of *NRN* and the serial number of the eID card can be combined as an index to store a part of the secret. Every citizen can –online– lookup his current and previous card numbers at the National Registry.

The *proxy certificate* mechanism allows owners of an eID card to setup mutually authenticated secure channels without the need for a trusted third party. Secure communication is even no longer restricted to SSL. The proposed system with proxy certificates makes it more flexible and extensible. Although not completely complying with the standards, the proposed scheme supports asymmetric *encryption* with the eID card. Moreover, the proxy certificates can be used for asynchronous communication (i.e. recipients can decrypt confidential messages after the communication channel is closed). Once the sender has deleted the eID certificate (as imposed by Belgian legislation), he can still check the validity of the proxy certificate and the revocation status of the eID certificate. Although other certificates in the validation path (i.e. *Citizen_CA*, *Belgium_Root_CA*, ...) may have been revoked, the proxy certificate can include the issuer of the eID certificate as an extra attribute to verify the validity of the rest of the certificate chain. Note that optional attributes in the proxy certificate may further restrict its use.

7 A Mobile and Secure e-mail Client

Both extensions discussed above are combined into a proof-of-concept application. An e-mail client has been developed. It supports exchanging confidential e-mail messages using the BeID proxy mechanism described in section 5. Individuals can use the e-mail service at any location as the necessary key material and contact information are stored securely on an easy accessible remote server.

7.1 Requirements

- R_1: The secure e-mail application is simple to use.
- R_2: The e-mail service is ubiquitously accessible.
- R_3: Certificates can be revoked (using CRL, OCSP).
- R_4: Contact information is kept private.
- R_5: Data stored on a remote server is useless to any third party.

7.2 Protocols

In this discussion, abstraction is made of the message format and the e-mail system that is responsible for the transport of e-mail messages. The protocols are designed to be compatible with existing e-mail systems.

setupProxy-Protocol (cfr. table 5). This protocol defines the creation and storage of an *e-mail* proxy certificate. The user U generates a proxy certificate with his e-mail address as an extra attribute (1). Next, a list of already existing credentials *creds* (i.e. certificates and corresponding private keys) is fetched from the remote secure store (2). The new credential consisting of SK_{proxy} and *proxyCert* is added to the credential list (3). Finally, the updated list is uploaded to the remote store (4).

Table 5. Creating and remote storage of a proxy

setupProxy():

(1) U	:	$(SK_{proxy}, proxyCert) \leftarrow \mathsf{createBeIDProxy}([E-mail : U.email])$
(2) U ← S	:	$creds \leftarrow \mathsf{retrieveSensitiveData}(U.email + ".creds")$
(3) U	:	$creds* \leftarrow \mathsf{addToCredentials}(creds, [SK_{Proxy}, proxyCert])$
(4) U → S	:	$\mathsf{storeSensitiveData}(creds*, U.email + ".creds")$

receiveBeIDProxy-Protocol (cfr. table 6) defines how a user $U1$ requests a valid *proxyCert* from another user $U2$ and adds it to his contacts in the remote secure

Table 6. Retrieve the proxy certificate of a contact

receiveBeIDProxy():

(1)	U1 → U2	:	requestCertificate()
(2)	U2 ← S2	:	$creds \leftarrow$ retrieveSensitiveData($U2.email + ".creds")$
(3)	U2	:	$proxyCert \leftarrow$ lookup($creds, U2.email$)
(4)	U1 ← U2	:	$(proxyCert, eID_2.cert_{sig}, cert_{Citizen_{CA}}, cert_{BelgiumRoot_{CA}})$
(5)	U1	:	if (!isValidCert($proxyCert, [eID_2.cert_{sig}, \ldots]$)) abort
(6)	U1	:	delete($eID_2.cert_{sig}$)
(7)	U1 ← S1	:	$contacts \leftarrow$ retrieveSensitiveData($U1.email + ".contacts"$)
(8)	U1	:	$contacts* \leftarrow$ addToContacts($contacts$, $[proxyCert, certChain/eID_2.cert_sig]$)
(9)	U1 → S1	:	storeSensitiveData($contacts*, U1.email + ".contacts"$)

store. Storing the contact certificates online is required to enable remote access from hosts on other locations. $U1$ requests the proxy certificate of $U2$ (1). Then, $U2$ fetches his certificate from his secure credential store (2-3) and sends it to $U1$ (4). $U1$ verifies the validity of the proxy certificate, taking into account the modified certificate path generation (5). To comply with the Belgian legislation, $eID_2.cert_{sig}$ is deleted after validation (6). Finally, $U1$ adds the proxy certificate to his secure contact store indexed with the e-mail address (7-9).

sendEncryptedEmail-Protocol. Table 7 shows the protocol to send a confidential message between two users. First, $U1$ fetches the proxy certificate $proxyCert_{U2}$, which he received earlier, from his contact store (1-2). $U1$ verifies the validity of the $proxyCert_{U2}$ (3). To improve the performance, a symmetric encryption scheme is used to encrypt the message. Therefore, a new random symmetric key sK is created (4-5). Next, the symmetric key sK itself is encrypted with the public key $proxyCert_{U2}.PK$ in the proxy certificate of the contact $U2$ (6). Both the encrypted message and encrypted symmetric key are mailed to $U2$ (7). To read the confidential message, $U2$ fetches the corresponding secret key SK_{U2} (8-9)) to decrypt the symmetric key $sK*$ (10). Finally, the message is decrypted with $sK*$ (11).

Table 7. Sending and receiving an encrypted message

sendSecureEmail(message):

(1)	U1 ← S	:	$contacts_{U1} \leftarrow$ retrieveSensitiveData($U1.email + ".contacts"$)
(2)	U1	:	$(proxyCert_{U2}, certChain) \leftarrow$ lookup($contacts_{U1}, U2.email$)
(3)	U1	:	if (!isValidProxy($proxyCert_{U2}, certChain$))abort
(4)	U1	:	$sK \leftarrow$ createSymmetricKey()
(5)	U1	:	$E_{data} \leftarrow$ encrypt(msg, sK)
(6)	U1	:	$E_{sK} \leftarrow$ encrypt($sK, proxyCert_{U2}.PK$)
(7)	U1 → U2	:	sendEmail($E_{data}, E_{sK}, U2.email$)
(8)	U2 ← S	:	$creds \leftarrow$ retrieveSensitiveData($U2.email + ".creds"$)
(9)	U2	:	$SK_{U2} \leftarrow$ lookupKey($U2.email, creds_{U2}$)
(10)	U2	:	$symKey* \leftarrow$ decrypt(E_{sK}, SK_{U2})
(11)	U2	:	$message* \leftarrow$ decrypt(E_{data},$symKey*$)

7.3 Evaluation

This paragraph first evaluates the initial requirements. Next, our solution is compared to other approaches for securing e-mail services.

- R_1: Only the setup and request of a proxy certificate may imply some additional user interaction. The other protocols can be processed transparently. Of course when accessing the remote secure store or creating new proxy certificates, two PINs of the eID card are required: for authentication and signing (key generation). This may seem awkward. However, a possible solution may be to detach the fetching and storing of secure data (i.e. retrieveSensitiveData resp. storeSensitiveData) from the protocols above and only perform them at the initialization and closing of the e-mail client: the user will have to enter his authentication PIN once and his signing PIN twice (of which one can be avoided if the application caches the signature temporarily).
- R_2: Confidential e-mails can be sent and received from any host with a card reader.
- R_3: A separate server maintains CRLs of revoked proxy certificates. Moreover, a sender can verify that the eID certificate of the owner of a proxy certificate is (not) revoked.
- R_4: Access to contact information is only possible using the Belgian eID. Adversaries do not have access to the proxy certificates of the contacts.
- R_5: All information stored on the remote secure storage server is encrypted and if the server is trustworthy (i.e. the server does not store the encrypted tag nor the *NRN*), an adversary cannot link any data or even discover the presence of data of a particular individual. Moreover, only the owner can get access to his encrypted data.

PGP [13] has a different trust model compared to our approach. PGP was initially based on chains of trust, while our solution is based on valid certificate chains. The trust level of public keys in PGP depends on the number of signatures on these keys by other users. Verifying the validity of those keys is not trivial and less reliable. Users have to interpret trust levels themselves. In the more recent OpenPGP specification [14], trust signatures can be used to support the creation of certificate authorities.

S/MIME [15] also offers a solution to send confidential messages based on certificate chains. S/MIME typically stores keys and certificates on a user workstation which implies that e-mail cannot be sent or retrieved on different hosts. Some attempts have been made to store keys and certificates for e-mail services on a remote server. Those solutions mainly use password based encryption to support ubiquitous access. Our eID based solution provides stronger security while maximizing the availability.

Moreover, the creation of a BeID proxy certificate does not require a complex registration procedure. The individual can even create a proxy certificate off-line. Some certificate authorities offer free e-mail certificates for exclusive S/MIME usage on their web site. Users must sign up for an account. However, this does not automatically allow usage of one's name in the certificate. For that, one has to prove ones identity in person to at least two Thawte notaries that are part of their Web of Trust.

PGP and S/MIME also support digital signatures and mechanisms to ensure the integrity of e-mails. Since the eID card can directly be used to sign e-mail messages, we have omitted the description of this functionality of our e-mail client.

8 Conclusion

This paper presents two reusable extensions to the Belgian eID technology, namely a *ubiquitously accessible remote secure storage service* and *a mechanism to issue proxy certificates*. The former allows Belgian citizens to manage sensitive personal data such as contact information, passwords, keys, tickets, etc. The latter can be used to self-certify asymmetric encryption keys. The validation of the proxy certificates slightly deviates from the standard, because of the current design of the eID certificates and restrictions imposed by the Belgian legislation. However, this problem can easily be solved by redesigning the eID certificates. Hence, this paper also offers some guidelines for countries that consider introducing an eID card in the future. As a proof-of-concept, both extensions have been incorporated in a secure e-mail client. The approach is more secure than PGP and avoids some key management problems in S/MIME.

Acknowledgements

This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven, the IWT-SBO project ADAPID and the IWT-Tetra project e-IDea.

References

1. De Cock, D., Wolf, C., Preneel, B.: The Belgian Electronic Identity Card (Overview). LNI, vol. P-77, pp. 298–301. Bonner Köllen Verlag (2006)
2. Verhaeghe, P., Lapon, J., De Decker, B., Naessens, V., Verslype, K.: Security and privacy improvements for the belgian eid technology. In: 24th IFIP International Information Security Conference (SEC). Springer, Heidelberg (2009)
3. Dumortier, J.: eID en de paradoks van het rijksregisternummer (2005)
4. Stern, M.: Belgian Electronic Identity Card content, 2nd edn., CSC, Zetes (2003)
5. Andries, P.: eID Middleware Architecture Document, 1st edn., Zetes (2003)
6. Rommelaere, J.: Belgian Electronic Identity Card Middleware Programmers Guide, 1st edn., Zetes (2003)
7. Ramlot, G.: eID Hierarchy and Certificate Profiles, 3rd edn., Zetes – Certipost (2006)
8. Belgian certificate revocation list, `http://status.eid.belgium.be`
9. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Rfc 3820 - Internet x.509 public key infrastructure (pki) proxy certificate profile (2004)

10. Housley, R., Polk, W., Ford, W., Solo, D.: Rfc 3280 - Internet x. 509 public key infrastructure certificate (pki) and certificate revocation list (crl) profile (2002)
11. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
12. Bellare, M., Goldwasser, S.: Verifiable partial key escrow. In: CCS 1997: Proceedings of the 4th ACM conference on Computer and communications security, pp. 78–91. ACM, New York (1997)
13. Garfinkel, S.: PGP: Pretty Good Privacy. O'Reilly Media, Sebastopol (1994)
14. Callas, J., Donnerhacke, L., Finney, H., Shaw, D., Thayer, R.: Rfc 4880 (Proposed Standard) – OpenPGP Message Format (2007)
15. Ramsdell, B.: Rfc 2633 – s/mime version 3 message specification (1999)

Filtering SPAM in P2PSIP Communities with Web of Trust

Juho Heikkilä and Andrei Gurtov

Helsinki Institute for Information Technology
Helsinki University of Technology
{Juho.Heikkila,Andrei.Gurtov}@hiit.fi
http://trustinet.hiit.fi/

Abstract. Spam is a dominant problem on email systems today. One of the reasons is the lack of infrastructure for security and trust. As Voice over IP (VoIP) communication becomes increasingly popular, proliferation of spam calls is only a matter of time. As SIP identity scheme is practically similar to email, those share the same threats. We utilized Host Identity Protocol (HIP) to provide basic security, such as end-to-end encryption. To provide call filtering, however, other tools are needed. In this paper, we suggest applying trust paths familiar from the PGP web of trust to prevent unwanted communication in P2PSIP communities.

The goal is to provide trust visibility beyond the first hop without requiring people to openly share private data such as contact lists. Since our distributed environment limits global solutions, our proposal bases on scale-free distributed nodes which provide service to the social trust neighborhood. We have implemented the service as a freely deployable stand-alone HTTP server, which can be either independent or a part of the P2P overlay. We have evaluated the performance of the path finding algorithm using the social network data from the PGP web of trust.

Keywords: p2p, social networking, trust, spam prevention.

1 Introduction

One of the most significant plagues of Internet today is spam, or unsolicited email. During the last ten years the amount of abusive email has grown from virtually zero to what is now estimated between 80 and 95 percent of email traffic [1][2] and to be in the order of 100 billion messages every 24 hours in June 2007 [3].

Today, spam filters are common in any aware environment, and they clear most of the garbage so users never see it. Still, some spam comes through and there are some false positives, not to mention the wasted bandwidth. Therefore the spam problem is not cured, but still remains, kept in the background with band-aids to ease the pain.

Thanks to broadband and mobile access, people are staying connected online more and more. As high-speed wireless and flat-rate tariffs spread, another form of communication is becoming popular, Voice over IP (VoIP) calls, or Internet Telephony. SIP, Skype, Messenger, and other services provide people with ability to talk to each other in voice or video for practically no added cost. This opens new opportunities to the spam industry. Traditionally calling people has involved direct costs per second, but if soon more and more people are reachable without such direct costs, the aspect of building automated recorded calling services to advertise things like Viagra and adult services can become plausible and profitable.

These unsolicited calls, nick named SPIT (Spam over IP Telephony), pose an even greater harassment to people than traditional unsolicited email (SPAM) does. There are two fundamental differences between these forms of communication. First, while emails go to the inbox and wait until the user reads them, a call is made in real time and needs acute attention, disturbing anything the user is doing at the moment. Secondly, the content of an email message in the inbox is usually readily readable by the server running the spam filter. The content of a call is not available until the call actually happens, so it cannot be prefiltered based on content until the call is passed and answered. Therefore, voice calls are much more intrusive and they cannot be filtered based on content[1].

Methods for preventing SPIT need to be developed before it becomes a problem, otherwise we end up in a similar situation as with email, trying to fix things already broken. We can see that if VoIP systems are not designed to prevent SPIT, it will likely become the next pervasive medium for spammers [4]. In the worst case for VoIP, this means people will stray away from the medium, or there will be an abundance of incompatible non-standard systems, none of which is common enough to draw spammer attention.

Even in Finland, where unsolicited advertising is rare and prohibited, there has been talk of cases where automated services have called traditional phones asking for permission to make advertisement calls. As such calls are made in traditional networks, we can assume they will become common in their inexpensive counterpart (even if, as with SPAM, we expect the majority to come from global actors).

In this paper we will look at a distributed SIP environment and using PGP-like grass roots approach of a social web of trust to filter out calls coming from outside or too far in the social neighborhood.

The rest of the paper is divided as follows. Section 2 gives an overview of our underlying P2PSIP system and issues of trust in distributed systems. Section 3 outlines the research problem of preventing SPIT. Section 4 describes the proposed solution on a conceptual level, and Section 5 looks deeper into the actual implementation. Section 6 presents the test data and performance results and Section 7 concludes the paper.

2 A P2P SIP System

Session Initiation Protocol (SIP) [5], is a text-based IETF protocol standard for establishing and managing multimedia sessions. SIP handles tasks such as

[1] It may be noteworthy that voice-mail counters these two arguments, yet filtering audio based content is much harder than for text.

negotiating connection parameters and media encoding. SIP User Agents (UA), are identified by a SIP Address of Record (AOR), an email-address-like identifier.

In traditional, non-distributed, SIP, the user agents (UAs) can connect directly to each other if they know the connection parameters for the other party. In practice, however, a set of various trusted servers, such as SIP proxies and SIP registrars, is required for smooth operation of the service. Since this model has created administrative burdens, points of failure and a lack of ad-hoc capability, several P2P SIP systems have emerged and the IETF is currently standardizing the data protocol in the P2PSIP working group [6].

The Networking Research Group at Helsinki Institute for Information Technology (HIIT) has produced a prototype for such a decentralized SIP system [7], which is built on top of Host Identity Protocol (HIP) [8]. HIP communication architecture provides the tools required for achieving the identifier-locator split, i.e., separates the identity of a host from its location, such as the IP address. HIP hosts are identified by a Host Identifier (HI), the public key of a cryptographically generated key-pair. These are in turn dynamically mapped to network locations, providing transparent multihoming and mobility in addition to confidentiality and self-authentication. Therefore, we can use HIP to provide more secure identities and confidentiality to used channels, than SIP alone can. While HIP won't completely remove bootstrapping risks in distributed environment, it provides secure connections to verified parties. The HIP implementation used in the prototype is HIPL [9], which is available for mobile Linux platforms, such as the Nokia Internet Tablets.

Figure 1 presents the architecture. The P2PSIP peers are the devices of end users. These contain the user application (or UA) and also a local P2PSIP proxy. Instead of using large centralized proxies, the functionality is distributed to small lightweight proxies which share the workload. Moving proxy functionality from network servers to end devices is the basic concept behind P2PSIP systems. Various P2PSIP designs vary in which parts of the system they focus on, but

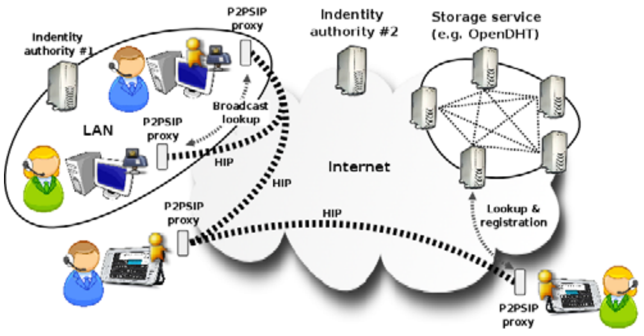


Fig. 1. The components of the P2PSIP system. In addition to users and peers, identity authority and overlay infrastructure can be present. The P2PSIP proxies handling the SPIT prevention are located in each end device. Peer-to-peer connections are made over HIP.

the local proxies forming a P2P overlay are common. Since the same proxy interface is still available, it has been possible to design the system to support legacy SIP applications.

The P2PSIP overlay network is not required for the UAs to make ad-hoc connections to each other, but it provides a global storage and an index. These services are essential for a smooth functioning of the system. The peers locate each other by the registrations stored in the overlay. Our implementation is not bound to a specific overlay, but can use multiple in parallel.

Finally, the Identity Authorities are centralized components for signing up for the system. These are trusted authorities which bind the SIP AOR to a public key using digital certificates, thus preventing identity thefts and spoofing. While Identity Authorities are centralized, there is no limit to their number, each providing service to their own domain, and once the registration and certification is done, there is no further need to stay in contact with the authority.

2.1 Trust in a Distributed System

We want to see how social trust relations could be used to filter calls. People seen as trustworthy by others in the social neighborhood should get through, while those seen in a negative light would be blocked.

Since our prototype is built on HIP and distributed SIP, the environment is by design distributed. This brings in new challenges, as we cannot rely on any centralized servers maintaining all the trust relations or reputation data. Balancing issues of privacy and reputation in a decentralized network is always a task requiring compromises, but our goal is to make the pay-offs greater than the costs.

In general, the question is a traditional introductory problem. If Ann wants to get in contact with Bob, but there is no direct link between them, Carol, who is acquainted with both Ann and Bob, can give Ann a reference to make her appear more trustworthy to Bob. However, in most cases we don't want to bother Carol explicitly, but only make it apparent to Bob that she sees Ann in a positive light. In more distant cases, where the trust chain is longer than two hops, a single introducer will not suffice. Further, unless Ann knows that Bob trusts Carol, she would be blind about knowing whom to ask for the recommendation. In a distributed network there is no node which would be guaranteed to know the best recommenders or shortest chains.

Even with limited knowledge we can make guesses and estimates. Without a comprehensive centralized database we cannot be certain we find the shortest or best link. If we do find a link, we will know that there is at least some link. While additional information could provide more trust, even a single link provides a basis for trust estimation. In case of purely positive reputation, it is also a guaranteed bottom estimate. Although we don't know if there are more available trust paths, we know those could only add to the estimate, not lessen it.

Social links can be used as credentials for contacting other people. People who are close in the social network are less likely to make SPIT calls. Just saying "Hi, I'm Ann, friend of Carol's" is likely to make Bob more receptive to talking to Ann in a traditional telephone call. This can be contrasted to email spam, however, where just shooting random names is commonplace. To be actually useful in an online environment, there needs to be some more proof that the link is real and that the person on the other end is who she claims to be.

There is a limiting factor, however. People's contacts are private information, and most people are not willing to publicize their contact lists. Even when there is nothing explicit to hide, people do not wish their employer to know their personal friends, or all their contacts to have access to the contact information of their children. Privacy of contact lists must therefore be protected.

While counter-claims say that in the age of Facebook and MySpace people no longer demand such level of privacy, it is fair to note that social networking sites are voluntary. People can choose which of their real links they reveal, as well as whether their friend list is available to the public, their friends, or nobody. In contrast to mandatory publication of telephone contact list by an operator, many of the most open-minded social network users would protest revealing such information. The ability to control information about oneself is one of the central concepts in privacy [10]. This control is not about enforcing fixed rules, but about managing and balancing based on the environment [11].

3 Beyond the First Hop

With the need to protect private data, how do we enable people to see the social network beyond the first hop (their own contacts). Although asking the contacts to share their contacts could be plausible in some limited cases, it is not really a solution which could be widely accepted. Also, adding hops in such a manner as to pass on other people's contact lists to achieve deeper visibility, would be a breach of trust and privacy.

Likewise, sharing the contact lists with a trusted centralized agent could solve the problem. However, such a solution is not applicable to distributed environment and further, would build a gigantic database of global contacts. The latter, while not commonly seen as a major problem today, can become such after the first major abuse of such database presents itself.

People should be able to choose who has access to their contact lists, and also, to which part of it. Naturally, managing contacts to hide or reveal details is an added burden for the user. However, it should be welcomed by anyone with even slight privacy concerns.

Further, since the goal is only to present credentials for the caller, there is no need to reveal the plain identities of the intermediaries. Therefore, obfuscating the identities should be a standard practice when passing contact lists to other parties. After all, the plain names have no meaning for the social trust structure, those who's real identity would provide any additional value are likely known and on one's own contact list to begin with.

Finally, making iterative searches over a distributed network is a time-consuming effort. If one needs to query data from various nodes, and especially nodes more than one hop away, the process is likely to take too long to process for an incoming real-time call. Making such searches will also burden the recipient node, (at least its network connection, which may be limited) making it vulnerable to Denial of Service (DOS) attacks.

We will present a method which provides trust information for nodes beyond one's own contact list, obfuscates user identities, hide identities of intermediary nodes to protect their privacy. The method places majority of the burden on the caller and on additional nodes which performance is not critical to the basic functionality of P2P SIP system.

4 Trusted Pathfinder Service

A Trusted Pathfinder is a service node in a flat network, which acts as a trusted third party to provide the call recipient with an anonymous trust path description of the caller. It fulfills this role by maintaining a protected and anonymized database of contact lists of people and finding trust paths in this web of trust. In a way it is similar to the web sites that find trust paths between PGP [12] keys, such as the PGP pathfinder [13] hosted by Utrech University.

To protect the privacy of the users, we have designed two obfuscating steps to minimize the exposure of private data. The first step is to hash all identities using SHA-1 at the client proxy. This will obfuscate the contents of the list even from the pathfinder service, preventing accidental exposure of administrators and requiring malicious admins to put at least some effort to decoding the identities. Hashing must not be confused with security, however. Secondly, the pathfinder will not reveal the intermediary nodes along the path, nor share the trust link data with any other parties. To preserve privacy of both private and business users, pathfinders must not share the contact data with each other.

The trust toward the Pathfinder is two-fold. First, the user who submits her (hashed) contact data to the service, should trust the service to keep her data private. Secondly, the user trusts that the path descriptions provided by the service are accurate to the extent possible with known data.

Note that both of these issues are in the human trust domain. This means that such issues can not be enforced by technical means, especially not in a peer-to-peer network. There is nothing to prevent a node administrator from running a service which appears legitimate, but leaks information. Submitting private data to a service ran by unknown actors is a risk and needs to be acknowledged as such. The P2P environment emphasizes the responsibility and decision-making of the user. Despite people's wishes, trust cannot be outsourced and there is no substitute for common sense and familiarity with the environment.

Our environment is distributed, so the pathfinder is not a global service either, but rather we see numerous local (in trust domain) services, and that the size distribution of the pathfinder nodes is likely scale-free [14]. Basically anyone could run their own pathfinder service; while the smallest ones would not have much public usage, it might be likely that circles of friends could run them, a company could run one for internal use, universities could run their own, etc. With client software modifications the user could further manage which contacts are shared with which pathfinder.

Since the pathfinders do not share their database with each other, a path query can be performed fast within the node without propagation delays. However, since the node does not contain global knowledge, the trust description it provides is limited. While this may sound like a limiting restriction, it may also model the trust network of people in a more human way. There is no need to be able to form paths with six degrees of separation, to every human on the planet. The goal is to reach beyond the first hop, the users' own contact lists. Since trust is not transitive, paths longer than three hops likely provide little introductory value. While the existence of any path could be interpreted as hint that the caller might not be a SPITter, infiltrating social networks is in practice easy as people find new contacts and adds only one to the length of the path. Adding a second 'cover' node adds two hops.

In the case of one additional hop, the friends of an infiltrating id could statistically find out who is responsible. If the path is longer, the question who has a friend with a spitting friend is very difficult to resolve without breaching the privacy of contact lists. Yet, the pathfinder itself, having internal access to the whole graph, could be modified to also run algorithms to locate regions linked by a single vertex, and consider those potential Sybil attackers similar to [15]. This functionality would be mostly useful in very large databases, however.

The pathfinder also acts as a small buffer against DoS attacks against the recipient. Since now the recipient proxy can require a path description before passing the call to the actual SIP client, the burden of pathfinding is on an external node. Certainly, the pathfinder can be taken down by an attack. However, since it is not a part of the P2PSIP infrastructure itself, but rather an additional service, removing it does not hinder the VoIP service. People who are one hop away (on a local contact list) can call freely without the pathfinder (though there may be other rules blocking the call). People can choose to accept calls without a path statement if they want. Attacking the pathfinder will, however, make life harder for those who are far, or not at all, in the social network, such as SPIT callers. Since direct contacts do not need the service at all, all delays posed by the system only apply to strangers.

Since the P2PSIP proxy is requiring the (unknown) caller to query the pathfinder service, all the delay in making the call can be put on the caller. Although, as we will show, the delay is not very long, it could be if the pathfinder is busy. Differentiating this waiting from a ringing tone should help a (human) caller perceive the call progressing. The recipient is not aware of this delay, his client starts ringing only after the call is passed through the proxy.

5 Implementation of the Pathfinder

We implemented the pathfinder as a stand-alone HTTP server written in Python [16]. The first goal was to test the performance need for tracing routes in the social network. A deployed call filtering service is likely to receive much higher load than the PGP pathfinder services on the web. While this is not an issue for small local servers, any nodes servicing a larger population may run into trouble if the workload is too high.

While we wanted a working solution and a proof-of-concept implementation to test, we didn't focus on optimization of the solution at this stage. There is likely still room for improvement in this regard for future versions of the implementation.

For the search algorithm we decided to keep to a basic two-way search [17], instead of putting an effort to tune a more complex algorithm. To optimize the search we built two data structures from the contact lists. The first contains the forward links as per the contact list. The other set contains reverse links readily available for the backward search.

The current implementation of the pathfinder provides the shortest trust path from the recipient of the call to the caller. This keeps the algorithm simple, as we do not need to keep track of the path or its properties. In future work we will consider adding metadata such as trust-weights to the links. Without such reservations for metadata we could have used Bloom Filters [18] for more secure obfuscation.

To begin, we add the recipient to a forward horizon, a list of nodes seen at a particular distance from the recipient, and mark the distance as zero. Then we add the caller to a reverse horizon and check if it matches the opposite horizon. If yes, we return the distance. As the caller likely is not the recipient, we increase the distance counter and return again to the other (forward) side. Then we start building a new horizon from unseen nodes on the contact lists of nodes on the previous, lower, horizon, and check if any of these match members of the opposing horizon. To prevent loops and to optimize the algorithm, we keep a record of nodes already seen, so we can ignore them in future. This process is iterated until the horizons meet, the maximum path length is reached, or all nodes are visited.

The algorithm, in a step by step manner:

- Take contacts from contact lists of a member of old horizon
- If these have not yet been seen, add to a new horizon
- If the new member matches the opposite horizon, return distance
- If the new horizon is completed and no match is found, increase the distance counter and iterate on the opposite side

The server also fits a contact list parser module, which interprets the contact list format(s) provided by user proxies to the database format used by the pathfinder. The current implementation supports our own XML schema, but a deployed implementation benefits from ability to understand various formats. As the data is never supposed to be exported, only import functionality is needed.

Finally, to provide integrity to the path description documents, a digital signature needs to be added to the response. For this purpose we add an S/MIME compliant signature to the document returned by the pathfinder server as a result of the query.

Figure 2 shows the modules in the implemented system as well as the processing paths for submitting contact lists and making path queries.

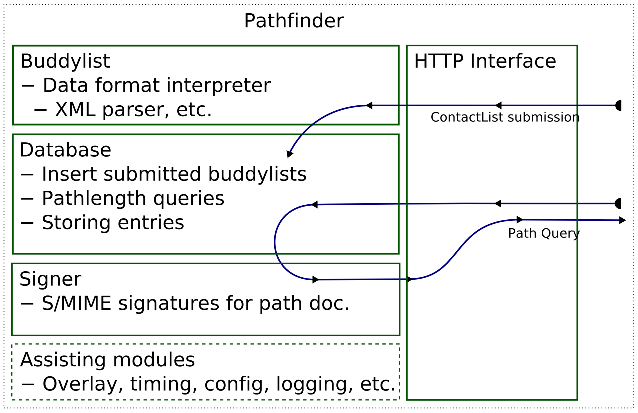


Fig. 2. The modules in the pathfinder system. The HTTP server accesses services provided by the database, XML-parser and signer, in addition to small help modules.

6 Performance

The pathfinder is now functional and can be used in conjunction with the P2PSIP system [19]. Although some performance issues remain, the median performance is suitable for small to medium scale deployment. We have a few optimization and scalability plans for the future, even as the current performance with the test dataset is satisfactory.

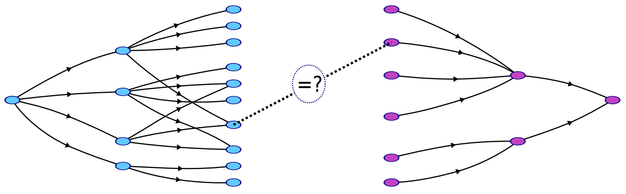


Fig. 3. The horizons in the social network are browsed by turns until they meet

6.1 Wotsap PGP Signature Data-Set

As a test dataset for the social network, we chose to use the Wotsap PGP key signature dataset available from the Wotsap website [20].

The dataset differs slightly from a real-world contact database since it only contains a strongly connected set of keys. In other words, each key in the set is reachable from every key in the set. There are no isolated nodes, or ones that have only incoming or outgoing links, unless we decide to insert them ourselves.

The links in the data-set are unidirectional, making it in similar to generic contact lists. Here, the topology differs from many social networks such as Facebook [21] or LinkedIn [22], but we feel that this approach is a better model for trust issues as trust is not generally symmetric.

The dataset contains about 35 000 keys and 350 000 signatures, giving an average of about 10 links and median of 3 links per node. While it can be argued that a realistic social network would be more dense, after consideration we decided that the dataset represents a generic social network in a sufficient manner for testing purposes.

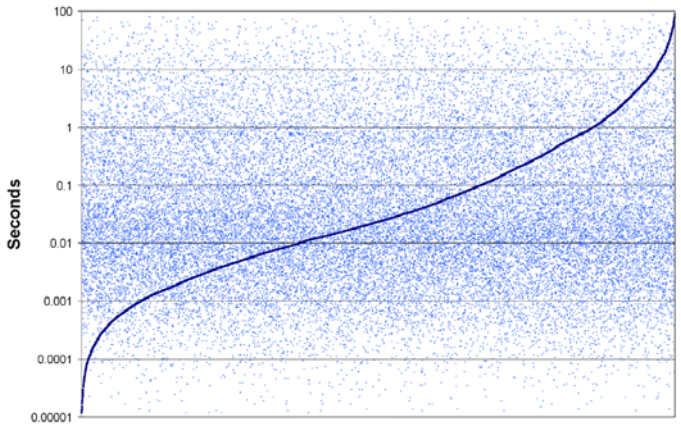


Fig. 4. The performance of pathfinder. Times of finding a path from a random node to a random node. 30 000 iterations shown in test order (cloud) and sorted by time (curve).

6.2 Measurements

We fetched 30 000 random-to-random paths using the pathfinder, setting the maximum path length at 6 hops. For real use such trust settings could be too long, but in the spirit of the six degrees of separation theory we wanted to see results for even longer chains. As noted above, our network is sparse, so not every chain fitted into these 6 hops. The measurements were made on an Athlon XP 3500+ single core PC.

The median time to find a path is 23 ms, which is quite applicable for lightly loaded servers, but of course insufficient for heavy loaded servers processing hundreds of requests per second. The deviation of the results is quite wide. While 10% of the cases take less than 1 ms, the highest 10% take more than 1.9 seconds to a maximum of 104 seconds.

In over 90% of cases the added delay is acceptable, but when the delay builds to the class of 3-5 seconds and more, it is likely irritable to a caller. Especially when considering possible HIP base exchange delays on low-power devices, as well as other network delays, informing the user on call set-up progress needs to be addressed in the design of the user interface.

The delay is not directly connected to the social distance, but the work done by the server. This can be seen from Figure 5 where the correlation between the path search delay and the number of unique nodes encountered in the process

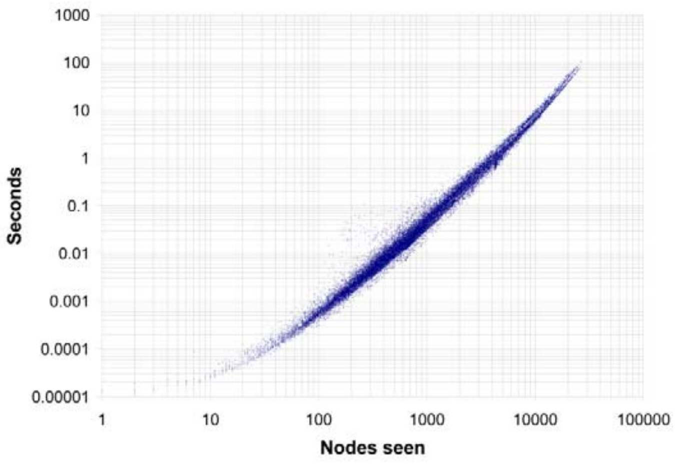


Fig. 5. The performance of pathfinder. Here the path finding time is presented in relation to the number of nodes seen on a loglog scale.

is shown on a loglog scale. We would also like to point that people who share regular contact with each other are likely be on each others' contact lists and thus should not be required to fetch path credentials when calling. Path search is a kind of introducer service, and the delay is an additional credential fetching cost which only applies to callers who are not on recipient's contact list.

7 Conclusion

We have designed and implemented a privacy preserving service for finding social trust paths between nodes which are further than one hop away from each other in the P2PSIP social network. Users submit their obfuscated contact lists to the trusted pathfinder service, which at request provides digitally signed credentials for the caller to present to the call recipient proxy. The pathfinders will keep the contents of user contact lists private, and will not reveal the path itself, only a description of it (path length in the current implementation). Large pathfinders could also try to locate likely Sybil attackers. We ran performance tests on the implementation and found it to be adequate.

References

1. Messaging Anti-Abuse Working Group: Email metrics program: the network operator's perspective, report #7 (2008)
2. spamhaus.org: Effective spam filtering, `http://www.spamhaus.org/effective_filtering.html` (Referenced: 2008-08-06)
3. spamunit.com: Spam Statistics, `http://www.spamunit.com/spam-statistics/` (Referenced: 2008-08-06)

4. Croft, N.J., Olivier, M.S.: A Model for Spam Prevention in IP Telephony Networks using Anonymous Verifying Authorities, University of Pretoria (2005)
5. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol, RFC 3261 (Proposed Standard) (2002)
6. IETF P2PSIP WG, http://www.ietf.org/html.charters/p2psip-charter.html
7. Koskela, J.: A HIP-based peer-to-peer communication system. In: Proceedings of the 15th International Conference on Telecommunications (2008)
8. Moskowitz, R., Nikander, P.: Host Identity Protocol (HIP) Architecture, RFC 4423, Informational (2006)
9. HIP for Linux, http://hipl.hiit.fi (Referenced: 2008-08-06)
10. Altman, I.: The environment and social behaviour: Privacy, personal space, territory, crowding, Brooks/Cole Pub. Co. (1975)
11. Palen, L., Dourish, P.: Unpacking privacy for a networked world. CHI Letters 5(1) (2003)
12. Zimmermann, P.: PGP User's Guide, Volume I: Essential topics (1994), http://www.pa.msu.edu/reference/pgpdoc1.html
13. Penning, H.P., Feisthammel, P.: PGP pathfinder and statistics, http://pgp.cs.uu.nl/ (Referenced: 2008-08-25)
14. Barabasi, A.-L.: Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life, Plume Books (2003)
15. Yu, H., Kaminsky, P., Gibbons, P., Flaxman, A.: SybilGuard: Defending Against Sybil Attacks via Social Networks. In: Proceedings of the ACM SIGCOMM 2006 (2006)
16. http://www.python.org/
17. Aura, T.: Fast access control decisions from delegation certificate databases. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, p. 284. Springer, Heidelberg (1998)
18. Wikipedia, Bloom filter, http://en.wikipedia.org/wiki/Bloom_filter (Referenced: 2008-08-14)
19. Koskela, J., Heikkila, J., Gurtov, A.: A secure P2P SIP system with SPAM prevention. In: Poster in Mobicom 2008 (2008)
20. http://www.lysator.liu.se/~jc/wotsap/ (Referenced: 2008-08-14)
21. http://www.facebook.com/
22. http://www.linkedin.com/

Generating Random and Pseudorandom Sequences in Mobile Devices

Jan Krhovjak, Vashek Matyas, and Jiri Zizkovsky

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xkrhovj,matyas,xzizkovs}@fi.muni.cz

Abstract. In our paper we study practical aspects of random and pseudorandom number generation in mobile environments. We examine and analyze several sources of randomness available in current mobile phones and other mobile devices at the application level. We identify good physical sources of randomness that are capable of generating data with high entropy in reasonable time and we investigate some relevant aspects (such as security, energy requirements, performance) of integrating selected pseudorandom number generators in the Symbian OS environment. The main contribution of this paper is the identification and analysis of randomness sources in mobile devices and a practical proposal for their post-processing, including a prototype implementation.

Keywords: mobile device, random sequence, source of randomness.

1 Introduction

Unpredictable cryptographic keys, padding values or per-message secrets are critical to securing communication by modern cryptographic techniques. Their generation typically requires an unpredictable physical source of random numbers and secure mechanism for their digital postprocessing.

Most common generation techniques involve truly random and pseudorandom number generators. The former are typically based on a nondeterministic physical phenomena (e.g., radioactive decay or thermal noise), while the latter are only deterministic algorithms where all randomness of the output is dependent on the randomness of one or several inputs (often called seed). Generation of pseudorandom data is typically (in most environments) faster and truly random data is used in this process only as the initial input.

Our paper deals with issues related to the generation of truly random and pseudorandom data (i.e., bits, numbers, and sequences) in mobile computing environments. Mobile devices are different from general purpose computers, and are now commonly used for security-critical applications like mobile banking, secure voice and data communication, etc. However, current mobile platforms do not provide a suitable built-in entropy source for seeding a pseudorandom generator. We consider the camera and the microphone noise as the most promising candidates for good sources of randomness. They provide a considerable amount of randomness (entropy) in a given time period and can be easily (with minimal postprocessing) used as a reliable true random number generator (TRNG).

In order to use random data for cryptographical purposes we need to obtain a sequence of unpredictable random bits (i.e., with a sufficient amount of entropy) distributed according to the uniform distribution. Since almost all external physical sources of randomness can be observed or influenced (gamma rays, temperature, etc.) resulting in non-random or even constant values, we postprocess resulting data with a pseudorandom number generator (PRNG)[1].

We implemented ANSI X9.31 (former X9.17) and Fortuna PRNGs for mobile devices with Symbian OS 9.x. Both these PRNGs are based on classical cryptographic primitives (e.g., AES) and use available randomness sources during the whole generation process. This property implies robust and secure design with capability of recovering from internal state compromise. Both our implementations are a proof of concept that demonstrates the feasibility of generating high-quality pseudorandom data in mobile phones at the application level.

A detailed discussion regarding identification, testing, evaluation of physical sources of randomness, and entropy estimation can be found in [6]. Some issues described in this paper were discussed at the conceptual level also in [7].

2 Requirements on Random Data

Let us begin with a basic description of requirements on random data for cryptographic purposes. We can distinguish between qualitative and quantitative requirements for random data. The former cover good statistical properties of generated random data and unpredictability of such data, while the latter deal with measuring of randomness and also cover demands of used cryptographic techniques and the performance issues of cryptographic generators.

2.1 Qualitative Requirements

Random data generated directly from a randomness source often contains statistical defects and dependencies causing parts of the random data to be easily predictable. These statistical defects are typically inducted by hardware generators during the sampling of the analogue randomness source or by influencing the sampled physical randomness source (e.g., by an active adversary).

The first step towards unpredictability lies in ensuring (at least to certain level) good statistical properties of generated random data. This can be partially solved by using digital postprocessing and/or statistical testing. Digital postprocessing is the deterministic procedure capable to reduce statistical relations and dependencies (including bias and correlation of adjacent bits). Statistical testing allows to (manually) detect some design flaws of a generator or to (automatically) avoid breaking or influencing the generator during its lifecycle.

[1] Note that an ideal PRNG for cryptographic purposes produces sequences that is unpredictable and computationally indistinguishable from the truly random data.

Typical techniques of digital postprocessing involve use of deterministic pseudorandom number generators or randomness extractors. The former serve only for spreading simple statistical defects into a longer sequence of bits. The latter allow to condense available input randomness to the most compact form that has uniformly distributed bits without statistical defects. Pseudorandom number generators as well as randomness extractors can be based on cryptographic primitives (e.g., hash functions) or simple mathematical functions. However, none of deterministic digital postprocessing methods can be used for improving initial randomness from the physical source.

The advantage of randomness extractors lies in good theoretical and mathematical background – more sophisticated randomness extractors can even provide some provable guarantees of the quality (resulting distribution) and quantity (in terms of extracted so-called min-entropy) of its output. We can distinguish deterministic or non-deterministic randomness extractors. The former work only on limited classes of randomness sources. The latter do not have this restriction, but they need an additional truly random input. Unfortunately, probability distributions of randomness sources must be in both cases precisely defined, otherwise it is not possible to construct appropriate and well working randomness extractor. In addition to that, the usage of an randomness extractor makes the generation of random numbers even slower. More details can be found in [11].

There are several commonly used statistical test suites or batteries (e.g., CRYPT-X, DIEHARD, and NIST) for verification of the statistical quality of random data by detecting deviations from true randomness (for details see [10]). However, no finite set of statistical tests can be viewed as complete and the results of statistical testing must be always interpreted with some care and caution to avoid wrong conclusions about a specific randomness source or generator.

2.2 Quantitative Requirements

A precise comparison of several sources of randomness requires also some measure of randomness. Basic measure is in information theory often called uncertainty or entropy and referred as Shannon entropy or alternatively information entropy. The well-known Shannon formula computes the entropy according to all observed probabilities of values in the probability distribution. This results only in average case entropy that is inappropriate for cryptography purposes. To (partially) cope with this situation the worst case min-entropy measure is often used. Min-entropy formula computes the entropy according to the most probable value in probability distribution.

Unfortunately, both entropy formulas have one serious drawback: they work only for exactly defined (and fixed) randomness distribution. In our case we cannot make any assumptions about distributions dynamically formed by used sources of randomness (e.g., they can be under ongoing attack) and this can always imply biased results in terms of entropy. This is a fundamental problem that can be seen also in the theory of randomness extractors. To partially cope with this problem, we perform all our experiments and entropy estimations in the worst conditions – i.e., under simulated attack on physical randomness source.

3 Randomness in Mobile Devices

Mobile devices are considerably different from general purpose computers and this also influences the process of generating random and pseudorandom data. The possibility to change the environment where a mobile device operates is definitely a great advantage given by the mobility nature of the device. The existence of several embedded input devices such as microphone, radio receiver, video camera, or touchable display is another advantage[2].

On the other hand, mobility and small physical size of devices bring also a higher possibility of theft or (temporary) loss with a potential compromise of the generator state. The important assumption of secure generator design is thus the impossibility of deducing the (previous/future) inner state of the generator and fast recovery of its entropy level (after a time-limited compromise).

Well-designed and robust generator must always have a sufficient amount of entropy – shortly after turning the device on, after letting the device out of sight, and after an intensive generation of random data. This non-trivial task may require employment of energetically costly sources of randomness (e.g., video camera) and/or the user contribution. Utilization of these sources may also be required to assure higher security – e.g., for mobile banking purposes.

3.1 Sources of Randomness in Mobile Devices

This part of the paper deals with practical experiments performed on two smartphones Nokia N73 with the Symbian OS and two similar PDA phones E-Ten X500 and E-Ten M700 with the Windows Mobile OS. The goal of our experiments was to assess the quality of selected sources of randomness in these mobile devices and to estimate the amount of randomness (entropy) in these sources. We used two identical Nokia N73 devices since we wanted to verify the correctness of our results or to detect unexpected behavior of the smartphone – in a case when one device suffers, e.g., by some manufacturing defect. Some issues described in this section were discussed and described in more detail in [6].

Due to the API restrictions in the Symbian OS, we were forced to drop sources of randomness like the battery level, signal strength or GPS position as measurements over these sources do not provide output (at the API level) with a sufficient precision (e.g., battery and signal values are available in the form of an integer between 0 and 10) or frequency (e.g., external GPS provides only one measurement per second).

On the contrary, microphone and digital camera perform a high-rate sampling of physical sources, yielding high volumes of data. Since we can never guarantee the quality of a physical source, our analysis is concentrated on the microphone and the camera noise that arises, e.g., in the CCD/CMOS chip or A/D converter, and is always present in the output data.

[2] We are aware that in general purpose computers use of audio/video data for generating random numbers is not a novel idea (see, e.g., [2]), but mobile devices with embedded cameras and microphones have a clear advantage with respect to fitness and practical applications of such techniques.

Microphone input: We wanted to evaluate all microphones in the worst possible conditions, therefore the first idea of our test settings involved recording of: some music sample, audio feedback, noise in an extremely quiet room. After several basic experiments with the built-in notebook microphone we saw that an audio feedback (i.e., sound loop between an audio input and an audio output) brings even more entropy than the music sample recording. Since the recorded noise is also present in music or other audio samples, the worst conditions for the microphone input arise from recording of noise in an extremely quiet environment (which was in our case a closed quiet room in the night).

We analyzed several microphones that have obviously different characteristic, e.g., due to different solidity of membrane or other manufacturing differences. 204 800 captured noise samples[3] were used to form a histogram of values. Furthermore, we used min-entropy (worst case) formula for quantitative analysis and for estimation of entropy in the generated data. Our min-entropy estimations, with the assumption of independency within the samples, are: 0.5 bits of entropy per sample for Nokia N73 hands-free microphone; 0.016 bits of entropy per sample for E-Ten M700 embedded microphone; 0.023 bits of entropy per sample for the E-Ten X500 embedded microphone.

As the next step we tested statistical quality and independency of captured noise samples. We used the Fast Fourier Transform for analyzing the basic frequency components present in the noise – ideal noise is expected to have all frequencies uniformly present. Several harmonic frequencies (narrow peaks in spectrogram) were revealed in spectrum of E-Ten devices and the best result (i.e., smoothest frequency spectrum) belongs to the Nokia N73 hands-free microphone with only few harmonic frequencies (see Figure 1).

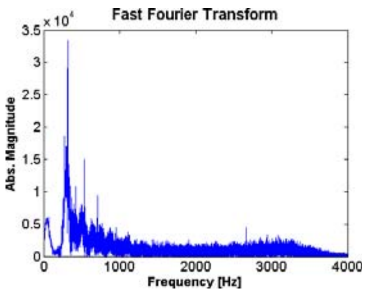


Fig. 1. Microphone frequency spectrum

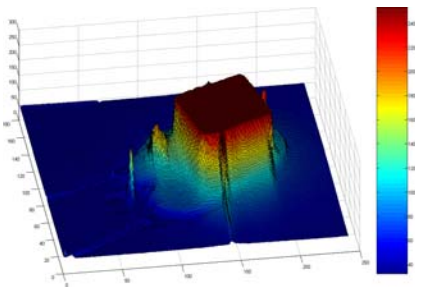


Fig. 2. Overexposure by a halogen lamp

We performed also several correlation tests that found a correlation in the recorded samples of noise. This correlation decreased as we took only every second/third value from our samples. Sequence created from every fourth value was without any statistically significant correlations. We therefore recommended

[3] We set 16-bit pulse coded modulation (a signed PCM) at the frequency 8000 Hz for sampling a sound wave.

to lower the estimated entropy at least 5 times with respect to the amount calculated for each sample value.

Camera input: Digital cameras for mobile devices can be based on several different silicon optical sensors, typically CCD or CMOS. All of them use an array of semiconductor photo-sensors to transfer an accumulated electric charge to a voltage. Low-quality sensors are typically used in mobile devices, and these sensors have a higher noise level than sensors used in standard digital cameras.

The worst conditions for these sensors involve recording of a static image (white or black background). The input source is often available even when the camera cover is closed or covered. This is both convenient and useful – it serves as a defense against an active attacker that illuminates the sensors and forces them to produce biased values. Illumination of the Nokia N73 CCD sensor by a halogen lamp is depicted in Figure 2 – in this case the central area of a sensor is forced to produce maximum values (225) and all noise is effectively removed.

Several components of the noise arising from optical sensors can be distinguished (shot noise, read-out noise, etc.), but we are interested in the overall noise. It is a well-known fact that the predominant component of this noise is the thermal noise and its actual level may depend on physical conditions – namely the temperature (for details see [1]). Therefore, we performed all practical experiments with the camera set at two temperature levels: 5 °C and 45 °C. We detected an inside decrease of the noise towards lower temperature, but the noise is still significantly present to provide enough entropy.

We captured all the test data from the viewfinder[4] rather than from a high-resolution picture. We are aware that the viewfinder image is downsampled from a full resolution of the optical sensor, but our primary intent was to overcome other software post-processing techniques (noise reduction, compression, etc.). However, performed experiments confirmed the presence of low level post-processing (details are not provided by the manufactures, similarly as for the downsampling algorithm) as, for example, correcting of light intensity towards the border of lens or automatic ISO level correction. All these proprietary techniques make the analysis and entropy estimation much harder.

At least 2000 noise samples (at temperatures 5–8 °C) have been always used to create a histogram of values for a particular color pixel. Our analysis confirmed that all three tested devices have clearly different camera chips and noise patterns. Our min-entropy estimates, with the assumption of independency within the pixels and frames, are: 3.2/3.3/3.9 bits of entropy per R/G/B color pixel for Nokia N73; 3.0/2.9/3.7 bits of entropy per R/G/B color pixel for E-Ten M700; 1.3/2.2/0.8 bits of entropy per R/G/B color pixel for E-Ten X500.

For statistical testing of the quality of noise samples we used the NIST test battery, auto- and cross-correlation functions in Matlab, and Fast Fourier Transformation. Our experiments revealed no statistically significant dependencies (significance level was set to 0.01) between neighboring pixels, rows, or even successive pixels in frames.

[4] The used resolution is at least 180×240 pixels and recording speed is within 10–15 frames per second.

Full description of all our experiments with Nokia N73 and E-TEN X500/M700 devices and other related technical details can be found in [6].

3.2 Secure Pseudorandom Numbers

The advantage of pseudorandom number generators lies in secure masking (smoothing) of statistical deviations caused by temporarily corrupted or influenced sources of randomness. If these generators behave correctly (in a secure computing environment), the resulting sequences have their statistical quality assured and the only critical point remains resistance to cryptanalysis.

The majority of multi-user and mobile computing environments cannot be considered as a secure computing environment (due to malicious admins/users, malware, possibility of device temporary loss). A careful generator design must thus take into account both forward and backward security after a compromise of its internal secret state. Forward/backward security means that an attacker with the knowledge of internal state of a generator is not able to learn anything about previous/future states and outputs of the generator. Forward security can be guaranteed by using one-way functions and backward security requires periodical refreshing of the internal state with additional truly random data.

The consequence of periodical refreshing is utilizing all available random samples – with the possibility of their gathering and accumulation (so-called pooling) even in the time when no generation is required. Such hybrid generators behave deterministically only between two successive reseedings.

We selected ANSI X9.31 and Fortuna pseudorandom number generators for our reference implementation – their basic properties are described below.

ANSI X9.31 PRNG: The ANSI X9.31 (former X9.17) PRNG uses in our setting only one cryptographic primitive, a block cipher. The NIST specification [8] suggests two implementations using the algorithm 3DES or AES. The main difference between these two block ciphers is the supported length of a block (3DES uses 64-bit blocks, AES operates on 128-bit blocks).

Let K be a 128-bit AES secret key, which is reserved only for the generation of pseudorandom numbers. E_K denote AES encryption (ECB) under the key K, V is a 128-bit seed value, and DT is a 128-bit date/time vector. I is an intermediate value, and the symbol $\oplus$ is the exclusive-or (XOR) operator.

Then a pseudorandom output R is in each iteration generated as follows:

$$I = E_K(DT), \tag{1}$$

$$R = E_K(I \oplus V), \tag{2}$$

$$V = E_K(R \oplus I). \tag{3}$$

The generator has no embedded pooling mechanism and the entropy gathering is performed on the fly. The critical part of generator's inner state is the secret key K that is expected to be stored securely – otherwise the forward security can be no longer guaranteed. Another part of inner state, the value R, is produced directly as an output[5]. Moreover, the DT vector is based solely on low-entropy

[5] The generator was originally designed for generation of single DES keys, therefore the output was expected to be secret.

date/time – this implies slow recovery after a state compromise and therefore weak backward security.

Fortuna PRNG: Fortuna [3] currently represents one of the most sophisticated generator designs with an improved and automated mechanism of pooling and extremely simplified (almost removed) process of entropy estimation.

Fortuna PRNG consists of three components. The first part is the generator itself, it takes a fixed-size seed and outputs an arbitrary amount (often restricted to 1 MB per single request) of pseudorandom data. The generator is the most primitive component of the Fortuna – a block cipher in the counter mode with some refinements. It is recommended to use AES with a 256-bit key and a 128-bit counter. The key and the counter form a secret internal state of the generator.

After every request another 256 bits of pseudorandom data are generated for a new encryption key. Periodical rekeying ensures not only backward security, but also forward security of the generator as it is infeasible to get previous output data after the key change even when the attacker knows the secret internal state of the generator. There is no reset of the counter and this property prevents short cycles that could occur due to repeating key values.

The second component is called accumulator and its main purpose is to collect and pool entropy from various sources of randomness. There are 32 pools, which are filled with data from one or several randomness sources in a cyclical fashion. This design ensures that random events from a single source are distributed evenly over the pools and an attacker controlling only some randomness sources cannot control all these pools. The accumulator is also responsible for periodical reseeding of the generator. Reseeds are numbered (1, 2, 3, etc.) and the pool P_i is used if and only if 2^i is a divisor of the reseed counter. Thus P_0 is used every reseed, P_1 every second reseed, P_2 every fourth reseed, etc. The only entropy estimate that must be done sets the minimal pool size necessary for reseeding – but it can be quite optimistic, e.g., 64 bytes for required 128 bits of entropy.

The seed file is last component of the Fortuna PRNG that serves as the storage of high-entropy data. It preserves the PRNG internal state and ensures that even after rebooting the generator can produce good pseudorandom data. Quite a simple concept in theory, but practical implementation is very difficult and depends extremely on the environment and platform (for details see [3]).

4 Integration into Symbian OS

In this section we summarize details of our implementation and integration of ANSI X9.31 and Fortuna PRNGs into the Symbian OS 9.x. We used the smartphone Nokia N73 for our practical experiments and performance tests.

4.1 ANSI X9.31 PRNG

We implemented the ANSI X9.31 PRNG as a simple GUI application[6] that is based on the AES encryption function and the SHA-1 hash function. SHA-1 is

[6] The source code is available at: http://sourceforge.net/projects/ansix931prng/.

an internal Symbian library function and the AES is implemented as a reference code provided by the IAIK Krypto Group AES Lounge ported to the Symbian OS by Philipp Henkel [5]. The generator itself is implemented in one class.

There are object attributes that preserve the generator secret state and that are necessary for the generator initialization – the most important is a disposable 320-bit entropy pool. It is implemented as two SHA-1 contexts and the initial entropy comes from 5 samples from the camera viewfinder, 20 audio samples from the microphone and the timing of each keystroke. Construction and initialization of the camera and microphone objects run asynchronously and the speed of data acquisition from these sources is 1495 KB/s. Since the estimated amount of entropy from these samples radically exceeds 320-bit, we can expect that the pool contains 320-bit of entropy. The pool is used only for following operations: first 128 bits are used as AES key material, next 128 bits fill in the seed and remaining 64 bits initialize first half of the date/time vector. As the generator is not reseeded during its runtime, it never recovers from a compromised state.

We improved the original X9.31 generator by initializing the first half of date/time vector by truly random data that remains fixed during whole generation process, while the second half is updated in every iteration by 64-bit of date/time. This is obviously a tradeoff between security and energy efficiency – better backward security can be achieved by using purely truly random data instead of low-entropy date/time, but the continuously running camera and microphone will drain the battery very soon. Another disadvantage is that the running camera is not accessible for other applications.

This implementation can be used in applications that require some random data at the start, but we do not recommend its continuous usage. The generator has in fact poor forward and backward security. In case of forward security, it is very easy to get previously generated random bits, if an attacker gets the secret state of the generator. We also implemented another version of ANSI X9.31 based solely on a one-way hash function (in our case SHA-256), which provides a better forward security for the mobile environment, as there are no problems with key management and secure key storage [4].

Backward security depends crucially on the entropy of the date/time vector. However, after a state compromise the length of DT is in fact reduced to 64 bits and we must expect that the attacker can predict a significant amount of DT vector bits. An optimistic assumption is that the attacker knows the time of the data request with a minute precision. Theoretical precision of the time in Symbian is in order of microseconds, but the analysis has proved that the time value is always divisible by 125. It reduces the number of possibilities to 480 000, and so approximately 18.87 bits of entropy. If the attacker can predict the time with second precision, then he has only 8000 possibilities, i.e., 12.96 bits of entropy. Cryptanalytic attacks on ANSI X9.17/X9.31 are discussed in [9].

The actual speed of our implementation is only 2.44 KB/s with the reference non-optimized AES implementation of encryption speed 75 KB/s. The average speed of implementation based on SHA-256 is 3.9 KB/s. The above informal security analysis and slow performance in mobile phones clearly suggests that practical usage of the ANSI X9.31 PRNG is not advisable.

4.2 Fortuna PRNG

In this subsection we describe the implementation of Fortuna PRNG on Nokia Series60 mobile devices with the Symbian OS 9.x[7]. As compatibility is not as straightforward as claimed by the vendors, it is worth noting we used the Nokia N73 smartphone, as there can be some differences when using other models.

Fortuna PRNG is designed to run continually, gathering entropy and servicing user requests for pseudorandom data. Therefore we decided to implement Fortuna by means of the Symbian OS *Client-Server Framework*. The server component is the application without a graphical user interface, implementing whole functionality of the Fortuna PRNG. The client component is the interface to the Fortuna PRNG, implemented as a dynamic-link library. This library with four exported functions allows to start Fortuna server, create new session with the running server, and perform several operations with an established sessions (e.g., requests the Fortuna server for pseudorandom data). The Fortuna server is a crucial part of the Fortuna PRNG, implementing all the functionality. It is a common Symbian OS application without a GUI, running on the background as a separate process, collecting entropy and servicing user requests. The generator component class consists of 256 bits long AES key, 128 bits long counter and two objects representing the cipher context and its key. The accumulator component of the Fortuna PRNG consists of the array of 32 objects – each object represents one pool containing entropy from random events. While the pools are parsable strings of unbounded length, it is more practical to implement the pools just as hash contexts.

The seed file component does not have its own implementing class. It tries to open the seed file in the `/Private` folder of the application. This folder is accessible only to this application and to processes with the capability *AllFiles*. When the seed file exists and contains at least 64 bytes of data, the generator is seeded with this data and the seed file is updated with the newly generated 64 bytes of pseudorandom data. If the seed file is not found (or is not long enough), the generator will not be seeded and user data requests will have to wait until the pools accumulate enough entropy to initialize the generator properly. The Fortuna class contains two object attributes dedicated to reseeding. A new 64-bit counter incremented at each reseeding and the time when the last reseeding was done. At least 100 ms delay is required between two successive reseedings.

We implemented 3 randomness sources: the keyboard (`CFKeyRandSrc`), the microphone (`CFAudioRandSrc`) and the camera (`CFCamRandSrc`). The first source gathers entropy from keyboard events. The randomness source starts waiting for the key events and when the key is pressed, the window server[8] generates the key event and sends it to our application. The key event data itself does not contain much entropy, therefore it is not used at all. Only the time (12.96 bits of entropy) of key event arising is sent to the appropriate pool. The second randomness source gathers entropy from the microphone device. According to our entropy estimation 1 KB sample contains 51 bits of entropy. The period and

[7] The source code is available at: http://sourceforge.net/projects/fortunaprng/.

[8] A server which manages screen, keyboard and pointer on behalf of client applications.

the exact amount of data taken should be precisely tuned to fulfill the goals of the target application. In our case the audio sample is taken every minute.

The last implemented randomness source gathers entropy from the camera device. As one frame from the viewfinder yields almost 100 KB, the data is spread over all pools. Strictly speaking, 3180 bytes of data is added to each pool. According to our estimations, this represents 11 080 bits of entropy for each pool. Although in theory it is possible to capture the viewfinder frames continually, it would limit the device user considerably. The camera has a significant power consumption and moreover it would be reserved for the Fortuna server only. Therefore, we capture only one viewfinder frame every minute.

We performed five battery life tests[9] with continuous utilization of different sources of randomness. Keyboard events have been monitored at all times, but no keys were pressed during measurements. The lowest speed of our implementation in the continuous capturing mode is 13.85 KB/s. The detailed results are summarized in Table 1 – the first column presents the average time of battery life and the second column then standard deviation of our measurements. Our

Table 1. Fortuna energy requirements – battery life

Sources of randomness (continuous sampling)	Avg. time [hh:mm]	Std. deviation [mm:ss]
Keyboard, microphone	17:27	8:34
Keyboard, camera	3:48	2:24
Keyb., cam., micr.	3:24	3:36

current Fortuna implementation captures both audio and video periodically just once a minute. In this case the battery is exhausted approximately after 3 days, 18 hours and 13 minutes. This is a non-trivial battery stress in comparison to 10 days, 11 hours and 1 minute of producing pure pseudorandom data without any entropy pooling. Energy requirements of our current reference Fortuna implementation are 2.78 times higher then energy requirements of implementation without any entropy pooling. For non-critical applications we thus recommend to capture data only once per 5 minutes – the battery is then exhausted approximately after 7 days, 11 hours and 34 minutes.

5 Conclusions and Future Work

In this paper we show a practical approach to random data generation for the mobile environment, providing a clear path to developers of many security-critical applications for the mobile environment. We investigated several possible ways of generating both random and pseudorandom data in mobile devices with Symbian OS. We start with the identification of available sources of randomness and assessment of their quality and performance. Our analysis showed that mobile devices have several good sources of randomness – we confirmed that at least the

[9] All measurements were performed with a Li-Pol battery type BP-6M (970 mAh).

microphone and camera noise contain a sufficient amount of entropy and thus can be reliably used as a good sources of truly random data.

Our work then lead to the investigation of the truly random data postprocessing with the use of pseudorandom number generators that are able to utilize all accessible sources of randomness in the generation process. We selected ANSI X9.31 and Fortuna pseudorandom number generators that we then successfully implemented and integrated into the Symbian-based smartphone Nokia N73.

We also want to point out that on top of the truly random data sources that we examined further, other sources like battery and signal level (and in some specific situations also GPS position tracking) are worth investigating. This can be done, for example, on the completely open-source Linux-based OpenMoko cellphone or at a higher granularity by using external devices (such as a cellular modem or GPS unit). Work with Symbian OS phones, however, can also be addressed by teams that have a lower-level (API) access to the devices than that available to us and other ordinary developers.

Acknowledgment. We acknowledge the support of the research project of Czech Science Foundation No. 102/06/0711 and of the FIDIS Network of Excellence.

References

1. Kennedy, J.: Digital camera fundamentals. Andor Technology, `http://www.andor.com/pdfs/Digital%20Camera%20Fundamentals.pdf`
2. Eastlake, D., Cybercash, S.C., Schiller, J.: RFC1750: Randomness Recommendations for Security. MIT Press, Cambridge (1994)
3. Ferguson, N., Schneier, B.: Practical Cryptography. John Wiley & Sons, Chichester (2003)
4. Gutmann, P.: Software generation of practically strong random numbers. In: Proc. of the 7th USENIX Security Symposium, pp. 243–257. USENIX Association (1998)
5. Henkel, P.: Port of Rijndael Block Cipher to Symbian OSs (2005), `http://www.newlc.com/AES-Encryption.html`
6. Krhovjak, J., Svenda, P., Matyas, V.: The sources of randomness in mobile devices. In: Proc. of the 12th Nordic Workshop on Secure IT System, pp. 73–84. Reykjavik University (2007)
7. Krhovjak, J., Svenda, P., Matyas, V., Smolik, L.: The sources of randomness in smartphones with Symbian OS. In: Security and Protection of Information 2007, pp. 87–98. University of Defence (2007)
8. Keller, S.S.: NIST-recommended random number generator based on ANSI X9.31 Appendix A.2.4 using the 3-key triple DES and AES algorithms. NIST (2005)
9. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic attacks on pseudorandom number generators. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 168–188. Springer, Heidelberg (1998)
10. Rukhin, A., Soto, J., Nechvatal, J.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. In: NIST Special Publication 800-22 (2001)
11. Shaltiel, R.: Recent developements in explicit constructions of extractors. In: Bulletin of the EATCS, pp. 67–95 (2002)

A Context-Aware Security Framework for Next Generation Mobile Networks

Matteo Bandinelli, Federica Paganelli, Gianluca Vannuccini, and Dino Giuli

Università degli Studi di Firenze
Department of Electronics and Telecommunications, via S. Marta 3, Florence, Italy
{matteo.bandinelli,g.vannuccini}@gmail.com,
{federica.paganelli,dino.giuli}@unifi.it

Abstract. The openness and heterogeneity of next generation communication networks are now highlighting more security issues than those of traditional communication environments. Moreover users' security requirements can often change in mobile communication environments, depending on the situation in which the user is immersed. Our objective is to define a context-aware security framework for addressing the problems of end-to-end security on behalf of end-users. Based on context data acquisition and aggregation features, the framework uses contextual graphs to define security policies encompassing actions at different layers of communication systems' architecture, while adapting to changing circumstances.

Keywords: context-aware security, multi-layer security policy, security context, contextual graph, next generation communication networks.

1 Introduction

The emerging vision from research in Next Generation Networks (NGN) is that of an All-IP network of heterogeneous networks, integrating different access technologies seamlessly with respect to end users. This approach implies to move from traditional vertical architectures towards the definition of a common architecture providing open interfaces for heterogeneous communication and application service providers [15].

The openness and heterogeneity of NGN communication networks are now highlighting more security issues than those of traditional communication environments. Roberts et al. [18] provide a brief analysis of challenges for security implied by NGN architectures. For instance, mobility and heterogeneity of user devices as envisaged by NGN architectures are likely to present physical security threats, since control and configuration of the devices are directly managed by end users and not by network security administrators. Another issue is how to provide end users with secure end-to-end communication in a heterogeneous network infrastructure, with rapidly changing security requirements related to user's mobile environment.

The term "context-aware security" is widely used to identify an emerging research field trying to cope with the above-mentioned issues by applying a context-aware system design approach [5],[13],[14]. Most existing works ([1],[5],[14],[20]) focus on context-based security policies for adaptive authentication and authorization services at the application layer. Such an approach is quite limited with respect to the issues of the upcoming NGN scenario, where a more comprehensive approach for enforcing security actions at different layers of the communication architecture is needed.

Therefore, our objective is to define a context-aware security framework capable of defining security policies with proper adaptation to changing circumstances in order to address the problems of end-to-end security on behalf of end-users. Adaptive security policies are here conceived as a set of security actions pertaining to different layers of the security architecture as defined by the International Telecommunication Union (ITU): application layer (network-based applications), service layer (basic connectivity, transport and added value services) and infrastructure layer (physical network nodes and communication links) [11].

We provide a definition of security context based on a non-exhaustive categorization of security context items (such as user location, device capabilities, access network type) and we adopt contextual graphs to model the decision process for defining multi-layer security polices adapting to changing context.

A significant contribution of this paper is the integration and composition of multiple context categories into contextual graphs in order to specify multi-layer security policies in the NGN scenario. With respect to the related research work, our aim is to propose a more general and extensible approach to face the complex security challenges envisaged by the NGN architectures. A possible implementation of the model into a NGN architecture is then proposed, by identifying proper functionalities to achieve the contextual-graph-based adaptation of security policies.

The paper is organized as follows: Section 2 discusses related work in context-aware security; in Section 3 we describe the main elements of the Context-aware Security Framework, namely the security context model, the security actions (by providing some examples for each security layer) and the use of contextual graphs for modeling context-based multi-layer security policies; Section 4 proposes an architectural model of the context-aware security framework; Section 5 concludes the paper and gives further research directions.

2 Related Work

While several works exist in the research field of context-aware computing and context-aware services in several application domains [7][17], only a few works have investigated the use of context aware computing models for designing adaptive security mechanisms.

Covington et al. [5] proposes a Context-Aware Security Architecture (CASA) enabling the design of security services which use security-relevant “context” knowledge to provide flexible access control and policy enforcement. CASA has been used to implement a Role-Based Access Control model based on the concept of “environment role” (i.e. environmental conditions that are relevant to access control).

Masone [12] proposes a Role Definition Language to describe roles in terms of context information. Other works present context-aware authentication and authorization policies for augmenting network security in Intranet environments [20] and in ubiquitous computing environments [1].

In [14] a model for context-based security policies specification based on the use of contextual graphs is proposed. The model has been applied for deducing context-aware authorization policies to be enforced in a pervasive environment.

Some recent works are beginning to investigate mechanisms to address security issues in mobile environments with such a broader perspective. In the IST MAGNET Project, a Context-aware Security Manager (CASM) is responsible for adapting security policies based on a security level determined according to two context categories (user location/scenario and device constraints) [13]. In [21] a context-aware security policy agent activates new security actions according to a change of context parameters (i.e. user preferences, power and location of the mobile platform).

With respect to these contributions, our security context model mainly refers to [14], and takes into account a wider range of context parameters, as described in Section 3.3. This contextual knowledge is exploited in order to define the most appropriate approach for enforcing security at the infrastructure, service and application layer.

3 Context-Aware Security Framework

The proposed Contex-Aware security framework aims at providing extensible mechanisms for defining and enforcing security policies in NGN environments. Extensibility is here intended as the capability of the system to cope with new security requirements which may be determined by new application domains, information services to be secured, available sources of context information and/or new security actions which can be enforced thanks to technological advancement and standards evolution. Extensibility of the proposed framework is mainly supported by the contextual graph-based modeling approach as discussed in [2].

Hereafter we describe the main elements of the Context-aware Security Framework: the Security Context, a classification of security actions and the approach for context-aware security policy definition based on contextual graphs.

3.1 Security Context

In the domain of context-aware security, a definition of security context is the one provided in [14]:

"A security context is a set of information collected from the user's environment and the application environment and that is relevant to the security infrastructure of both the user and the application."

Referring to communication environment, as the set of users, devices, applications, data and communication links that characterize the communication, we propose the following definition:

"A security context is any information that can be used to characterize the security situation of a communication environment. The security situation can be characterized in terms of possible security threats, user status and requirements with respect to information protection, available communication and computing resources which can be exploited by a system in order to specify and enforce proper actions to guarantee end-to-end security".

With respect to the first definition our definition does not mention any possible context source and/or context information category (e.g. user and application environment), as most appropriate context information and related sources can vary according to application purposes and the available technological infrastructure. Moreover, as the focus is on end-to-end security, the context should consider not only the situation concerning the environment of a target end-user, but also the situation of the destination peer (i.e. the host delivering information services, or another user interacting via communication services with the first one). Based on the security context definition proposed above, we provide an enumeration of categories for organizing security context items. The enumeration does not aim at being exhaustive. Indeed, our attempt has been to select first those context items that can be considered of general value for securing information exchange and, possibly, can be acquired by common and easily deployable sensing infrastructure. In section 3.3 we also provide an example of how these context categories can be applied to define context-aware security policy in an application scenario.

We first defined the following basic context information categories:

- **User.** This category includes information items representing current user personal sphere, surrounding environment attributes, preferences and attitudes. For instance, the user environment can be characterized by multiple mobile and fixed devices available for accessing different kind of communication services. The following items can be included in this category:
  - *User Location*: symbolic position of the user. Possible values are: "public" (the user is located in a public environment, where possible eavesdroppers may be close to the user accessing the network), "private" (the user is located in private environment – e.g., a house, an office, where eavesdropping may be feasible only within the core network or via side-lobes of the private wireless network).
  - *User Status*: user's current activity and/or availability to interact with communication and information services. It can be associated to the "user status" typical of messaging services. Possible values are: "not available", "available", "only textual interaction", "only voice interaction".
  - *Available Devices*: devices that are associated to the target users and can be used to access network-based services. Examples of user devices are: mobile phones, Personal Digital Assistants (PDA), notebooks, desktop PCs, fixed phones.
  - *Required Security Level*: security level which is required by the end-user. Possible values are: "high" (the user is asking for a high security communication, this may be the case for a remote access to very critical data), "medium" (the user is asking for a medium security communication, this may be required for an access to personal, but not critical data), "low" (the user is not particularly asking for a specific security level, this may be the case for entertainment online services).

Many of the above mentioned items can be directly acquired by user manual input. However, different mechanisms can be put in place in order to automate the process of context information acquisition. For instance, positioning techniques can be integrated in user devices in order to localize users (e.g. GPS, RFID readers, etc.). Typically, localization mechanisms can acquire information on user's physical position (e.g. latitude and longitude coordinates). Proper physical-symbolic positions association can be defined for frequently visited locations (e.g. "home" and "work" locations). In order to estimate user required security level from available contextual information, several mechanisms could be alternatively or jointly applied, such as machine learning and rule-based inference mechanisms (for instance, a rule may state that the required security level is low when the user location is of type "private").

- **Device.** This context category aims at representing most relevant technical characteristics of the user "active" device, i.e. the device that the user is currently using to access network-based services to be secured. We selected the following basic context information:
  - *Device type*: possible types of end-user devices are, for instance: mobile phone, Personal Digital Assistant (PDA), notebook, desktop PC.
  - *Device capabilities*: it refers to the capabilities of the end-user devices, in terms of hardware, software and configuration settings. These capabilities can be classified as follows: hardware (es. CPU, memory), display (es. size, resolution), protocols (es. HTTP, SIP, SMTP), networking (es. Bluetooth, WI-FI, GPRS, UMTS), application platform (es. JVM).
  - *Security settings*: This includes parameters such as antivirus signature version, personal firewall availability and rules setting, routing tables, and file-system permissions settings.
  - *Available Energy Supply*: this parameter takes into account device resources which are relevant to energy management. It can be represented as a combination of two components: a flag variable accounting if the system is powered by an external AC (alternating current) input; a second parameter representing the remaining battery life.

This information on the active device can be useful in order to estimate security risks (for instance if the antivirus version is not up-to-date) and to evaluate if a desirable security action can be enforced on the target device, according to its capabilities. For instance, on a mobile device characterized by limited computational resources and high energy consumption, the system cannot enforce the activation of a VPN to secure the communication channel. Context items characterizing the device category can be represented by means of the composite capability/preference profile (CC/PP) representation scheme [19].

- **Communication**: here we include the characteristics of the end-user device communication link.
  Most relevant attributes are:
  - *Current access network type*: possible values are GPRS, UMTS, Wi-Fi, etc.
  - *Available bandwidth*: bandwidth available for information transmission. Possible symbolic values are: "high" (e.g. more than 2Mbps) "medium" (e.g. between 2 Mbps - 56 Kbps) "low" (e.g. 56 Kbps as traditional modem).
  - *Signal quality*: in case of wireless access network, this parameter represents the quality of the transmission signal. Possible values are "high, medium, "low". Mapping with physical values is dependent on the technological infrastructure and application domain.

- *Access network security*: security settings of the access network in use (e.g. encryption available/not available). This information may be useful for determining possible security risks (e.g. if the communication channel is not encrypted) and for selecting the set of possible security actions that can be reliably supported.

- **Application:** this context category includes parameters describing end-user applications' characteristics, such as:
  - *Content Sensitiveness*: content to be exchanged can be "sensitive" (e.g. user personal data) or "not-sensitive" (e.g. broadcast news).
  - *Content Size*: size of content to be exchanged. Possible symbolic values are "high", "medium", and "low". Mapping with physical values (e.g. kB) is dependent on the application domain.
  - *Active Application*: type of network-based application to be secured. Applications can be distinguished in user-to-user interaction (e.g. voice over IP, instant messaging, etc.) and access to information services (e.g. Web browsing).
  - *Currently Executing Applications*: currently executing applications that can have conflicts of interest with the target active application.
  - *User Application Profile*: it represents the user's role in a service application domain (e.g. patient, nurse, general practitioner in a e-health scenario). At lower-level, it can also be represented by a profile of permissions for accessing application services. In such case, possible values are "read only" (e.g. only "get" operations), "read & write" (e.g. only "get" and "post" operations), "superuser" (e.g. any operation)..

3.2 Security Actions

The objective of the proposed security framework is to exploit system's knowledge of context to select appropriate security actions to be enforced in order to guarantee an expected security level.

With respect to the state of the art [1],[14],[20], our work aims at providing a flexible framework which can encompass heterogeneous security actions (not just encryption, authentication and authorization), and, in principle, all the available tools and technologies capable of improving the protection of the exchanged data.

More precisely, our objective is to adopt a multi-layered approach, i.e. to exploit contextual knowledge in order to adapt security policies. Such policies are defined as a combination of actions at the different layers of the security architecture for systems providing end-to-end communications, as defined by the International Telecommunication Union (ITU).

The Security Layers are hereafter defined [11]:

- **Application Layer:** this layer focuses on security of the network-based applications accessed by end-users. Examples are: file transport and web browsing applications, voice messaging and email, as well as vertical applications (e.g., customer relationship management, electronic/mobile-commerce).
- **Service Layer:** this layer addresses security of services provided by service providers. Examples are: basic transport and connectivity services (e.g., AAA

services and domain name services), value-added services (QoS, VPN, location services).

Infrastructure Layer: This security layer focuses on the hardware resources at both network and end users' sites. Examples are: routers, switches and servers, communication links, end-user devices.

Table 1. Security Layers and Related Actions

Security Layer	**Security Action**	**Security Dimension**	**Security Action strength**
Infrastructure Layer	Device switching (vertical handover)	Availability, Communication Security	Cellular phone, PDA, notebook, desktop PC
Services Layer	MAC layer Encryption activation	Confidentiality	WEP, WPA (shorter key), WPA (longer key)
	VPN IPSec activation	Confidentiality, Authentication, Data Integrity	3DES, AES-128bit, AES-256bit, etc.
	Blocking of other concurrent session potentially harming the desired session	Communication Security	
Application Layer	SSL session activation.	Confidentiality, Authentication, Data Integrity	Plain SSL with not-trusted certificates, SSL with trusted certificates.
	User alerting	Privacy	Textual alert, audio alert, Video alert
	Data encryption	Confidentiality	Encryption algorithm (e.g. AES) with different key sizes and rounds number)
	Adaptive authentication	Authentication	Anonymous, Pseudonymous (weak auth), strong authentication

For each security layer, Table 1 shows a set of possible security actions (selected as most meaningful for the considered NGN scenarios). These actions can be enforced with different strength, depending on implementation choices. A possible, non exhaustive, list of implementation options is shown for each selected action.

With respect to the ITU-T X.805 standard specification, the security actions considered in this section address the following security dimensions: Authentication, Data Confidentiality, Communication Security, Data Integrity, Availability, and Privacy. The other Security Dimensions (Access Control, Non repudiation) have not been taken into account. As a matter of fact, such dimensions are already addressed by other works [14], which can complement or extend the functionalities offered by the proposed Context-aware Security Framework.

3.3 Context-Based Security Policies

As described in Section 2, our work is based on the approach proposed in [14]. In particular, [14] specifies that rule-based formalism is often adopted for modeling context-based decision processes, but it suffers from the following main limitations: the difficulty to maintain such formalisms in case of complex systems to secure, to identify all the needed contextual information from the rule-based formalism, and to understand the followed strategy of the policy. Therefore, we also adopted the contextual graph approach, which can be used to represent a set of practices to perform in order to solve a given problem. A path, from the input to the output of the contextual graph, represents a practice with the contextual elements explicitly considered.

A contextual graph is an acyclic graph with a unique input, a unique output and a serial-parallel organization of nodes connected by oriented arcs. A node can be an action to be enforced, a contextual node, or a recombination node. A contextual node represents the instantiation of a context item, which is evaluated in order to direct the process through one path among all the possible alternatives. A recombination node allows to represent the convergence of different paths into a single node. An action node represents a security action to be enforced.

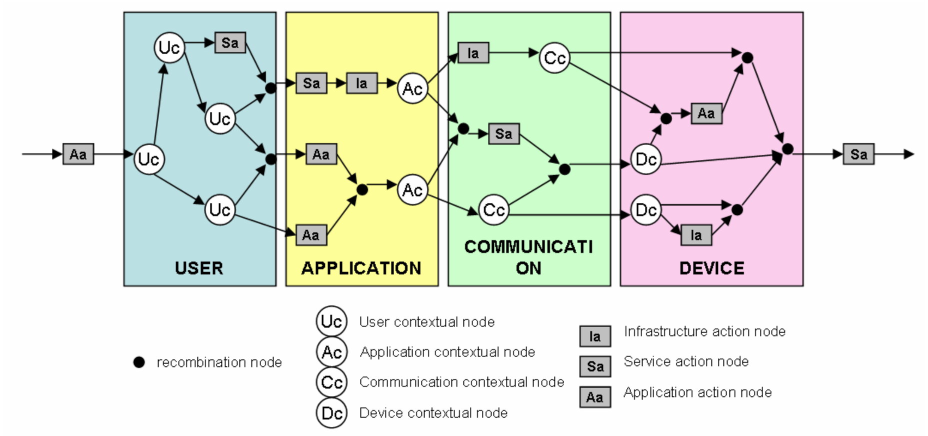


Fig. 1. An example of Contextual Graph

With respect to the Moustefaui proposal in [14] that has the aim to specify authentication and authorization policies, we extend the contextual graph theory, in order to cope with both the security categorization and the multi-layer approach of the security policy. As shown in Fig. 1, the decision process is based on the evaluation of the four categories of the security context and it is represented by a sequence of User, Application, Communication, or Device contextual nodes. Furthermore, a more complete end-to-end security policy is here represented by the instantiation of three different action node types (Infrastructure, Service and Application action nodes), in order to model a multi-layer security policy. Security context items which have been acquired by the system are evaluated in the graph at the corresponding contextual node, to select proper security actions, which are represented by action nodes. The contextual graph is thus a sequence of context evaluation nodes representing the most relevant attributes of the communication environment, and security enforcement nodes defining the security policy to be implemented in that specific communication environment.

In the following we describe a reference application scenario, used to provide an example of application of the contextual graph for the enforcement of multi-layer security policies.

At 8.00 am John, a doctor, is at the railway station, on the way to visit a patient (Mr Smith), who has been visited last week by a colleague. John decides to discuss Mr Smith's health record with his colleague. By accessing the open public WLAN available at the railway station, John uses his PDA to call the colleague (Situation A). During the conversation, John shares with the colleague the patient health record's information. John goes directly to the patient's home to visit him; then he accesses the information services of the hospital and update the patient record (Situation B).

In Fig. 2 we provide an example of how user, application, communication and device-related context can be taken into account for the definition of multi-layer security policies. For the sake of brevity, the example has the objective of presenting some relevant decision steps, not a complete decision path. Situation A is characterized by the following decision path, that represents the security context of the communication environment. First, John accesses the service network through a public access point at the railway station; due to his public location, he is exposed to greater threats then other locations. Second, the service is about transmitting sensitive content (the patient health record) and can be characterized as an interaction with another person. The access network is an "open" WLAN (i.e. link without encryption). As encryption is enabled on the user device (and also decryption is enabled at destination side) the content is encrypted before transmission. In Situation B, the user is accessing a service, which implies the access to sensitive content. As VPN is enabled at both user and destination site, the security action to be enforced is the VPN setup.

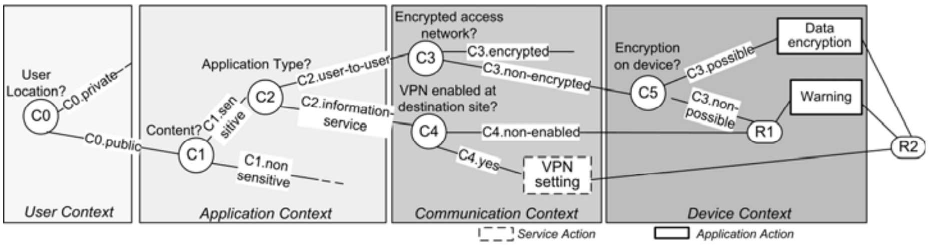


Fig. 2. Example of context-aware security policies

4 Architectural Model

This section describes how the considered security-context aware system can fit into a high-level architecture for adaptive security in Next Generation Networks. In Figure 3 we depict the main functional blocks of the architecture.

Context Data Acquisition: This block collects context information about the communication environment, related to users, devices, applications, data and communication links. Context data are acquired by heterogeneous context providers,

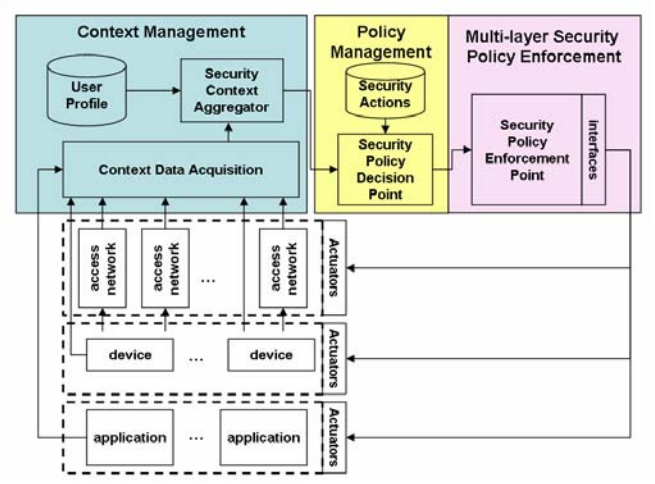


Fig. 3. Architecture of the Security Context Aware Framework

by exposing an interface, or, more precisely, a composition of different interfaces for push/pull data acquisition. For instance, device information may be gathered by the device Management Information Base via SNMP, or CC/PP messages. Information related to the end user may be collected either from the user profiles managed by the network infrastructure, via network signaling protocols, or by application-dependent profile information, through remote interface invocations (e.g. via Web Service interfaces).

Context Data Aggregator: It integrates and correlates raw context data collected by the Context Data Acquisition in order to obtain context data at the level of abstraction needed in the decision process and to maintain data consistency and quality. Several data processing and reasoning techniques can be adopted for this purpose: a programmatic approach, rule-based reasoning, ontology-based reasoning, self-learning techniques, etc.

Security Policy Decision Point (PDP)**:** This block defines the proper multi-layer security policy according to the current context. This component builds the above-described contextual graph and searches for a path matching with available context items' values. Security actions that are selected along the path are communicated to the Security Policy Enforcement Point for their actuation.

Security Policy Enforcement Point (PEP)**:** This block takes as input the context-based security actions to drive their enforcement, by communicating with the appropriate actuators situated in the network nodes and/or in the end-user device. As security actions may involve different layers (as depicted in Fig. 2) the Security PEP interacts via proper interfaces with different actuators.

Actuators: The actuators are active entities listening for commands sent by the Policy Enforcement Point and communicating with software and hardware resources for the realization of security actions. While the decision process can be conceived as a centralized module in the considered architecture, the security policy actuation is typically distributed, due to the extreme heterogeneity of the possible enforcing points and actuators (e.g., current device, alternative devices, network access point, etc).

Since it is expected that the NGN also will use the enhanced IP Multimedia Subsystem (IMS) to integrate IP-based multimedia services over the Internet, wireline telecommunications networks, and diverse wireless networks [16], in the following a possible implementation of the context-aware security framework in IMS is proposed. It should, indeed, be noted that the above-mentioned functional blocks are basically standard components that are already embedded in the whole IMS architecture.

The NGN architecture, according to the access independent principle, is characterized by multiple access networks converging into a common IP based core network. In such scenario, IMS provides specific gateways to the control plane for each access technology (e.g., GPRS, xDSL, WLAN, etc.), through specific Proxy Call State Control Function (P-CSCF). Furthermore, the suitability of the IMS as the environment in which our model can be implemented, results from the use of a wide set of Internet Engineering Task Force (IETF) protocols, in order to harmonize with Internet services. Specifically, IMS uses the Session Initiation Protocol (SIP) [10] for signaling and session management; Diameter [5] protocol and common open policy service (COPS) [8] for operations and management. Each of these protocols can represent a possible, although not complete, implementation of functionalities of our proposed model. Session Description Protocol (SDP [9]), which is used for describing multimedia sessions in order to support a suitable resources reservation, contains parameters (e.g., required bandwidth, service type, connection parameters, etc.) that are also useful during the decision process of our model and that can be used by Context Acquisition and Aggregation functions in order to derive many security context attributes. The user presence attributes can be derived from the IETF presence model [6], which is integrated in IMS through the use of an extension of the SIP protocol, called SIP/SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) [3]. Other user attributes, such as the addressing of the available devices, and access state parameters can be derived directly from the Home Subscriber Server (HSS) of the the IMS architecture. The main data stored in HSS include, indeed, user identities (the private one, and the public ones), registration information, access parameters and service-triggering information [16].

Also regarding the enforcement functions, their implementation could be driven by exploiting proper mechanisms native in NGN architecture, such as session mobility between different devices. According to our approach, the device switching, supported by the control plane of NGN architecture for QoS requirements [15], should be driven also by security issues. Further, relating to the enforcement of security actions, COPS protocol provides the suitable communication in a PDP-PEP mechanism, in order to manage the resource allocation for the current communication.

For other specific exchanges between user devices and network nodes inside a NGN architecture, extensions of the SIP protocol could be implemented [3].

Then, in such a NGN architecture, the decision process of our context aware security model is implemented by a dedicated service node of the control plane, or, referring to IMS architecture, by the Serving Call State Control Function (S-CSCF), which represents the central element of the signaling network. The adoption of the

IMS architecture as an enabling environment for the implementation of the proposed model is supported by the consideration of IMS as the reference platform for NGN control plane, however our context aware security model preserves his validity regardless of IMS, and allows possible implementation in other policy decision and enforcing systems.

Relating to a possible implementation of the Context-Aware Security in IMS, the sequence diagram proposed in Figure 4 refers, for example, to a session set-up, based on the request/response mechanism of a SIP architecture. With respect to a typical SIP interaction [10], the S-CSCF intercepts the communication between the SIP user agents, while activating the decision process and promoting the implementation of the consequent security policy, by exchanging “ACTION REQUEST” messages with enforcement points, here represented by the Control Plane Gateways (i.e., Proxies according to the user view) and by the SIP user agents on the user devices.

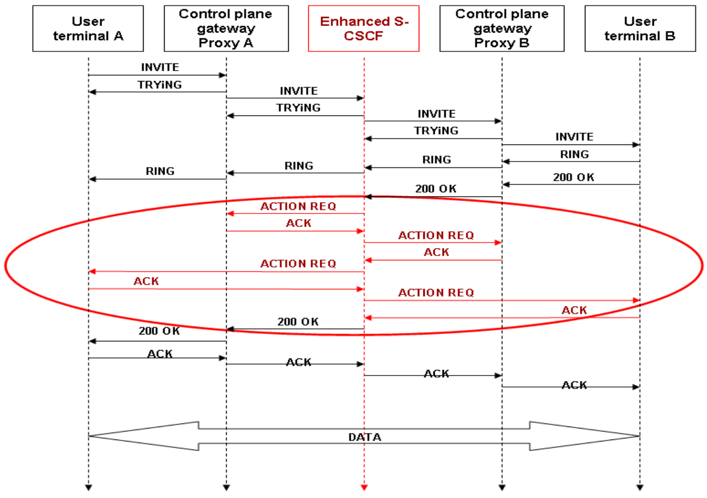


Fig. 4. Session set-up in a Context-Aware Security system in IMS

5 Conclusions

This work presented a context-aware security framework for addressing the problems of end-to-end security on behalf of end-users in a Next Generation Network scenario. We discussed the definition of context in the security domain and proposed a set of categories for organizing context items. The security framework uses contextual graphs to define security policies encompassing actions at different layers of communication systems’ architecture (i.e. application, service and infrastructure layers of the ITU-T security architecture), while adapting to changing context. In order to demonstrate the feasibility of the proposed system, we also discussed a possible implementation in a generalized NGN architecture. Future research activities will be devoted to the design and development of a prototype for the Security Context Aware Framework in an e-health application scenario. The analysis of the target scenario will also provide useful hints for refining the context items' and security actions' models. Furthermore we will investigate how the proposed approach could be extended in order to manage security policies at different levels of abstractions and/or priorities (e.g. by using sub-graphs and/or a hybrid approach integrating further reasoning techniques). This analysis will aim at defining high-level policies (e.g. matching with organizational and user requirements) and binding them with low-level implementation policies.

References

1. Al-Muhtadi, J., Ranganathan, A., Campbell, R., Mickunas, M.D.: Cerberus: a context-aware security scheme for smart spaces. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, March 23-26, pp. 489–496 (2003)
2. Brezillon, P., Pasquier, L., Pomerol, J.C.: Reasoning with contextual graphs. European Journal of Operational Research 136(2), 290–298 (2002)
3. Brok, J., Kumar, B., Meeuwissen, E., Batteram, H.J.: Enabling New Services by Exploiting Presence and Context Information in IMS. Wiley Interscience, Hoboken (2006)
4. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter Base Protocol. Request for Comments 3588 (September 2003), `http://www.ietf.org/rfc/rfc3588.txt`
5. Covington, M.J., Fogla, P., Zhan, Z., Ahamad, M.: A Context-Aware Security Architecture for Emerging Applications. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC), Las Vegas, Nevada, USA (December 2002)
6. Day, M., Sugano, H., Rosenberg, J.: A Model for Presence and Instant Messaging. Request for Comments 2778 (February 2000), `http://www.ietf.org/rfc/rfc2778.txt`
7. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. In: Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems, CHI 2000 (2000)
8. Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., Sastry, A.: The COPS Protocol. Request for Comments 2748 (January 2000), `http://www.ietf.org/rfc/rfc2748.txt`
9. Handley, M., Jacobson, V.: SDP: Session Description Protocol. Request for Comments: 2327 (April 1998), `http://www.ietf.org/rfc/rfc2327.txt`
10. Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J.: SIP: Session Initiation Protocol. Request For Comment 2543 (March 1999), `http://www.ietf.org/rfc/rfc2543.txt`
11. International Telecommunication Union. ITU-T X.805 Security architecture for systems providing end-to-end communications SERIES X: Data Networks and Open System Communications – Security (2003)
12. Masone, C.: Role definition language (rdl): A language to describe context-aware roles. Technical report, Dartmouth College, Computer Science, Hanover, NH (2002)
13. Mihovska, A., Prasad, N.R.: Adaptive Security Architecture based on EC-MQV Algorithm in Personal Network (PN). In: 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Philadelphia, USA (2007)

14. Mostefaoui, G.K., Brezillon, P.: Modeling context-based security policies with contextual graphs. In: The Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 28–32 (2004)
15. Nasser, N., Hasswa, A., Hassanein, H.: Handoffs in Fourth Generation Heterogeneous Networks. IEEE Communications Magazine (October 2006)
16. Poikselka, M., Mayer, G., Khartbil, H., Niemi, A.: The IMS – IP Multimedia Concepts and Services. Wiley, Chichester (2006)
17. Ranganathan, A., Campbell, R.H.: An Infrastructure for Context-Awareness based on First Order Logic. Personal and Ubiquitous Computing 7(6), 353–364 (2003)
18. Roberts, M.L., Temple, M.A., Mills, R.F., Raines, R.A.: Evolution of the air interface of cellular communications systems toward 4G realization. In: Communications Surveys & Tutorials, vol. 8(1), pp. 2–23. IEEE, Los Alamitos (First Quarter 2006)
19. World Wide Web Consortium, Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0 (January 2004)
20. Wullems, C., Looi, M., Clark, A.: Towards Context-aware Security: An Authorization Architecture for Intranet Environments. In: The Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications, pp. 132–137 (2004)
21. Yee, G., Korba, L.: Context-aware Security Policy Agent for Mobile Internet Services. In: Proc. of the 2005 IFIP International Conference on Intelligence in Communication Systems (2005)

Information Reconciliation Using Reliability in Secret Key Agreement Scheme with ESPAR Antenna

Takayuki Shimizu, Hisato Iwai, and Hideichi Sasaoka

Doshisha University, 1-3 Tatara, Miyakodani, Kyotanabe city, Kyoto, 610-0394 Japan
etj1101@mail4.doshisha.ac.jp, {iwai,hsasaoka}@mail.doshisha.ac.jp

Abstract. As a countermeasure for eavesdroppers in wireless communications, a secret key agreement scheme using a variable directional antenna called ESPAR antenna was developed. In this scheme, the process of information reconciliation is necessary to correct the discrepancies between the legitimate users' keys. In this paper, we propose a new information reconciliation protocol using the reliability of the raw keys. The proposed information reconciliation protocol is a modified version of the protocol used in quantum key distribution called Cascade. The results of simulations show that the proposed protocol can correct errors with less the number of disclosed bits and less the number of communications than those of Cascade.

Keywords: key agreement, ESPAR antenna, information reconciliation, Cascade.

1 Introduction

As wireless communications become increasingly pervasive, they are becoming targets for attacks. Unfortunately, it is easy for an adversary to eavesdrop wireless communications because of the broadcast nature of wireless communications. To establish secure communication links in wireless communications, two types of cryptographic techniques, public-key cryptography and symmetric-key cryptography, are generally used. In wireless LAN systems, the symmetric-key cryptography such as AES [1] is used because of its high processing speed. However, the symmetric-key cryptography has two operational issues: key distribution and key management. Moreover, since cryptographic techniques used currently are based on unproven assumptions regarding the hardness of computational problems, they would be broken if the assumptions were to be debunked.

Meanwhile, information-theoretic cryptography has been developing [2,3,4] as a cryptographic technique for which security is proven even against an adversary with unbounded computing power. There are several information-theoretically secure cryptographic techniques such as one-time pad [2] and secret communications over a noisy channel [3,4]. To share one-time pad keys, an information-theoretically secure key agreement scheme from correlated source outputs was proposed [5,6]. Quantum key distribution [7] is also known as an information-theoretically secure key agreement scheme.

Recently, novel techniques utilizing radio propagation characteristics for secret key agreement have been developed [8,9,10,11] as information-theoretically secure key agreement. They exploit the reciprocity of radio propagation characteristics, which provides similar radio propagation characteristics for two communicating terminals, while it is difficult for an eavesdropper at a different location to obtain the similar characteristics because of the locality of the radio propagation characteristics. This is a good approach to solve the issues of key distribution and key management because they can provide a one-time key whenever it is needed. In particular, a secret key agreement scheme using an electronically steerable parasitic array radiator (ESPAR) antenna [12] is more effective for environments where the fluctuation of the radio propagation characteristics is slow such as indoor wireless LAN environments [9].

In the secret key agreement scheme using the ESPAR antenna, secret keys are generated based on received signal strength indicator (RSSI) profiles measured by two legitimate users. The generated raw keys, however, may contain some errors (the discrepancies between the legitimate users' keys), which are caused by the noise and other effects at the receivers. In this situation, the legitimate users need to eliminate or correct the errors by public discussion, while the amount of the information revealed in the public discussion to eliminate or correct errors is as little as possible. This process is called information reconciliation. So far, various information reconciliation protocols have been proposed [7,9,13,14,15]. Among them, Cascade [13] is representative one and is widely used in quantum key distribution because of its high performance being close to the theoretical limit on the number of exchanged bits.

In this paper, we propose a modified version of Cascade using the reliability of the keys in which RSSI information is used as the reliability to correct errors effectively. We carry out numerical simulations, and the results of the simulations show that our information reconciliation protocol is suitable for the secret key agreement scheme using the ESPAR antenna.

2 Secret Key Agreement Scheme Using ESPAR Antenna

In this section, we recall the secret key agreement scheme using the ESPAR antenna [9]. The assumed conditions are as follows. There are two legitimate users, an access point (AP) equipped with an ESPAR antenna and an user terminal (UT) equipped with an omni-directional antenna. There is also an eavesdropper equipped with an omni-directional antenna. AP and UT can communicate with an identical frequency by using a method such as time division duplex (TDD).

Figure 1 shows the procedure of the secret key agreement scheme. The procedure consists of three steps: (1) randomness sharing, (2) information reconciliation, (3) privacy amplification. In the randomness sharing, AP and UT

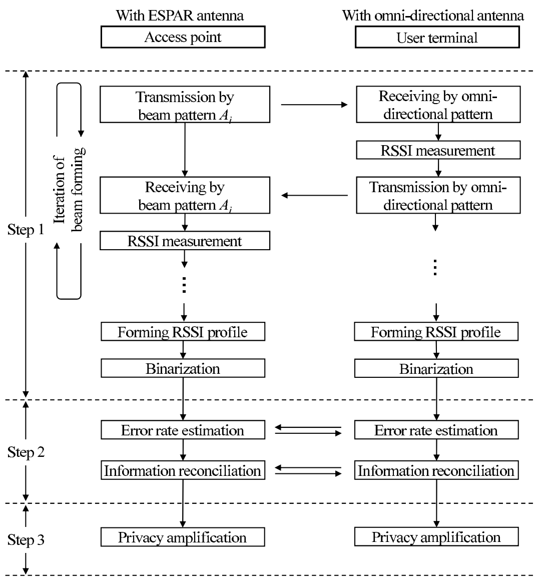

With ESPAR antenna
Access point
With omni-directional antenna
User terminal
Step 1
Iteration of beam forming
Transmission by beam pattern A_i
Receiving by omni-directional pattern
RSSI measurement
Receiving by beam pattern A_i
Transmission by omni-directional pattern
RSSI measurement
Forming RSSI profile
Forming RSSI profile
Binarization
Binarization
Step 2
Error rate estimation
Error rate estimation
Information reconciliation
Information reconciliation
Step 3
Privacy amplification
Privacy amplification

Fig. 1. Procedure of secret key agreement scheme using ESPAR antenna

generate key candidates from their RSSI profiles. Firstly, one terminal transmits one packet and the other measure RSSI alternately, where the beam pattern of the ESPAR antenna is fixed. Then, the beam pattern of the ESPAR antenna is switched randomly by changing the reactance values of the ESPAR antenna. This process is repeated until the sufficient length of RSSI profiles is obtained. After measuring the RSSI profiles, AP and UT binarize their RSSI profiles in order to generate key candidates, where the median of each RSSI profile is used as the threshold for binarization.

In the information reconciliation, AP and UT correct the discrepancies between their key candidates by public discussion. To eliminate the errors, AP and UT remove RSSIs around the threshold, where the positions to be removed are sent over a public channel because such the RSSIs are susceptible to the noise. Furthermore, the remaining errors are corrected based on the syndromes of the key candidates by applying error correcting techniques, where the syndrome of UT is sent over the public channel. To minimize the amount of the information leaked to the eavesdropper, AP and UT estimate the bit error rate of the key candidates by generating and disclosing dummy keys [16], and they set appropriate parameters of the information reconciliation protocol such as the length of elimination and an error correcting code.

In the privacy amplification, AP and UT generate a secure secret key from the identical key candidates by diminishing the partial information leaked to the eavesdropper. The privacy amplification can be realized by using universal hash functions [17]. Thus, AP and UT share an identical secure secret key.

3 Cascade Protocol

In this section, we recall Cascade protocol [13]. The conventional information reconciliation protocol using error correcting codes [9] may introduce additional errors if the number of errors is greater than the error correcting capability, and the reconciliation fails. In contrast, Cascade never introduces additional errors, and the number of errors decreases exponentially by iteratively applying Cascade.

Before describing the algorithm of Cascade, we explain BINARY, which plays the role in detecting and correcting the discrepancies of the legitimate users' keys in Cascade. An example of BINARY is shown in Fig. 2. When the legitimate users' keys have an odd number of errors, AP and UT perform an interactive binary search to find and correct an error in the following manner.

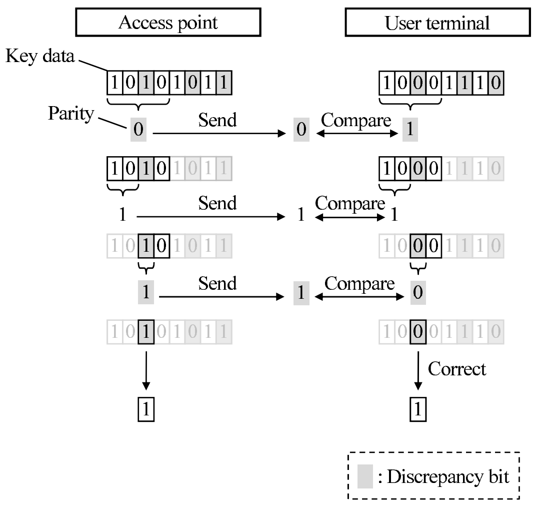


Fig. 2. Example of BINARY

BINARY:

1. AP and UT calculate the parity of the first half of their key candidates.
2. AP sends UT the calculated parity.
3. UT determines whether an odd number of errors occurred in the first half or in the last half by comparing the parity of UT to that of AP.
4. The steps 1∼3 are repeatedly applied to the half block determined in the step 3 until one error is found. Finally, the found error is corrected.

Let us now describe the algorithm of Cascade. Cascade proceeds in several passes. The number of passes is determined by the legitimate users before execution. AP and UT perform Cascade in the following manner.

Cascade:

1. AP and UT permute their key candidates in random order.
2. In the pass i, AP and UT divide the permuted key candidates into blocks of k_i bits and calculate the parities of all the blocks.
3. AP sends the parities of all AP's blocks to UT.
4. By using BINARY, UT corrects an error in each block whose parity differs from that of AP's corresponding block. At this point, the blocks of AP and UT in the pass i have an even number of errors.
5. In the pass i (> 1), let l be the position of the bit corrected at the step 4. AP and UT find that the blocks including l in the passes $1 \sim i-1$ have an odd number of errors. Let $\mathcal{A}$ be the set of the blocks including l.
6. AP and UT choose the smallest block in $\mathcal{A}$ and use BINARY to find and correct another error.
7. Let l' be the position of the bit corrected at the step 6 and $\mathcal{B}$ be the set of the blocks including l' in the passes $1 \sim i$. AP and UT find that the blocks of the set $(\mathcal{B} \cup \mathcal{A}) \backslash (\mathcal{B} \cap \mathcal{A})$ have an odd number of errors.
8. AP and UT update the set $\mathcal{A}$ according to $\mathcal{A} \leftarrow (\mathcal{B} \cup \mathcal{A}) \backslash (\mathcal{B} \cap \mathcal{A})$.
9. The steps 6$\sim$8 are repeatedly applied until $\mathcal{A} = \emptyset$, at which the pass i ends. At the end of the pass i, each block in the passes $1 \sim i$ has an even number of errors (perhaps zero).

In Cascade, almost all errors are corrected until the end of the pass 2 when the appropriate block length is set in each pass [18]. The block lengths k_1 of the pass 1 and k_2 of the pass 2 minimizing the number of disclosed bits are as follows:

$$k_1 = \left\lfloor \frac{4 \ln 2}{3 p_{\mathrm{e}}} \right\rfloor \tag{1}$$

$$k_2 = \left\lfloor \frac{4 \ln 2}{p_{\mathrm{e}}} \right\rfloor, \tag{2}$$

where p_{e} denotes the bit error rate of the keys. In the case of k_i $(i \geq 3)$, the block lengths have not been optimized. The original Cascade [13] sets k_i $(i \geq 3)$ as follows:

$$k_i = 2k_{i-1} \quad (i \geq 3). \tag{3}$$

In practice, we estimate the bit error rate p_{e} by using dummy keys and set the block lengths according to Eqs. (1) $\sim$ (3).

4 Cascade Using Reliability

In this section, we propose a modified version of Cascade using the reliability of the keys, which is suitable for the secret key agreement scheme using the ESPAR antenna. When we apply the conventional Cascade to the secret key agreement using the ESPAR antenna, BINARY is not the best method to search and correct errors because BINARY searches errors with binary search using only binary key data and does not utilize RSSI information. Let K_x and K_y be the random variables of AP's key and UT's key, respectively. In this case, the conventional Cascade is considered hard-decision information reconciliation, and therefore AP and UT need to exchange at least

$$n_{\min}^{\text{hard}} = nH(K_x|K_y) = nH(K_y|K_x) \tag{4}$$

of information on average for their reconciliation by the Slepian-Wolf theorem [19], where n and $H(A|B)$ denote the key length and the conditional entropy of A given B, respectively.

In the proposed protocol, BINARY is replaced with error correction using the reliability of the keys in which we utilize RSSI information as the reliability measure to search errors effectively. Let R_x and R_y be the random variables of AP's RSSI and UT's RSSI, respectively. In this case, the proposed protocol is considered soft-decision information reconciliation, and therefore AP and UT need to exchange at least

$$\begin{aligned} n_{\min}^{\text{soft}} &= nH(K_x|K_y, R_y) \\ &= nH(K_x|R_y) \quad (\because H(K_y|R_y) = 0) \\ &= nH(K_y|R_x) \\ &\leq n_{\min}^{\text{hard}} \end{aligned} \tag{5}$$

of information on average for their reconciliation. Equation (5) shows that we can correct errors with less the number of disclosed bits by utilizing RSSI information than that of the conventional Cascade.

Before describing the algorithm of the error correction using the reliability, we define the reliability of the keys by taking into account a feature of key candidate generated from RSSI profile. Figure 3 shows examples of the bit error rate with regard to each bit of key candidates of 1024 bit sorted in ascending order of RSSI when SNR (Signal to Noise Ratio) is 20 dB and 30 dB. It can be seen from this figure that the bit error rate of key bits near the threshold of binarization, i.e. the median, is higher because key bits near the threshold are more susceptible to the noise. This figure allows us to conclude that key bits near the threshold of binarization have lower reliability. By taking into account the fact, we define the reliability α_i of the i th key bit as

$$\alpha_i = |r_i - r_{\text{th}}|, \tag{6}$$

where r_i and r_{th} denote the i th measured RSSI and the threshold of binarization, respectively.

Having defined the reliability, we will now explain the algorithm of the error correction using the reliability. An example of the error correction using the reliability is shown in Fig. 4. When the legitimate users' keys have an odd number of errors, AP and UT perform an interactive reliability-based search to find and correct an error in the following manner.

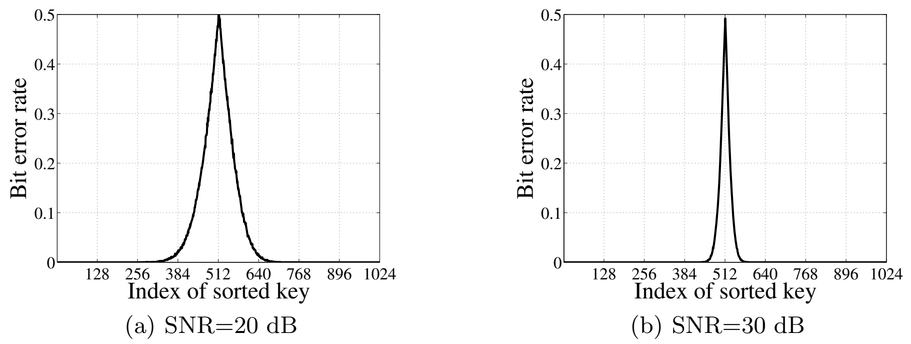


Fig. 3. Examples of bit error rate with regard to each bit of key candidates sorted in ascending order of RSSI

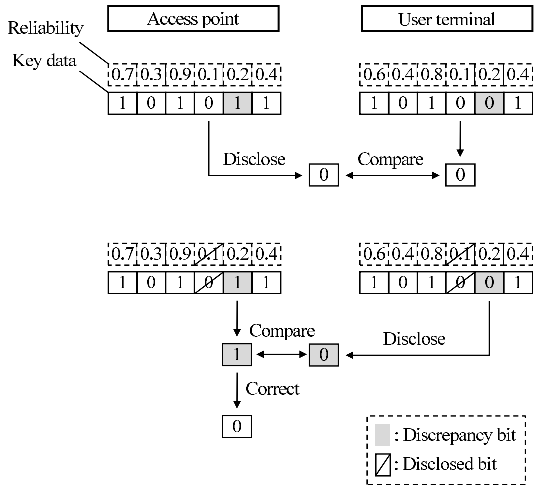


Fig. 4. Example of error correction using reliability in proposed protocol

Error correction using reliability:

1. AP sends the least reliable bit among the undisclosed bits to UT.
2. UT compares the bit sent by AP to the corresponding bit of UT. If the compared bits are different, UT corrects the error. If the compared bits are equal, UT sends the least reliable bit among the undisclosed bits to AP.
3. AP compares the bit sent by UT to the corresponding bit of AP. If the compared bits are different, AP corrects the error. If the compared bits are equal, AP sends the least reliable bit among the undisclosed bits to UT.
4. The steps 2$\sim$3 are repeatedly applied until one error is corrected.

5 Simulation Setting

5.1 System Model

We developed test equipment using ZigBee^{TM} [20], which conforms to IEEE802.15.4, of the secret key agreement scheme using the ESPAR antenna [9]. Therefore, we carry out numerical simulations assuming a model of IEEE802.15.4 to evaluate the performance of the proposed protocol. There are AP equipped with an ESPAR antenna and UT equipped with an omni-directional antenna. We assume that the eavesdropper is a passive attacker, i.e. she only listens to communications between AP and UT and does not interfere into them.

5.2 Simulation Model

Figure 5 shows the environment of the simulation. There are AP and UT in the same room enclosed by concrete walls on four sides in which there are no reflecting and scattering objects. The size of the room is 10 m $\times$ 8 m. Throughout the simulation, the position of AP is fixed to the center of the room, (0.0 m, 0.0 m), and the position of UT is fixed to (3.0 m, 2.0 m).

The parameters of the simulation are specified in Table 1. AP uses a 7-elements ESPAR antenna. The reactance vector of the ESPAR antenna is set randomly at every RSSI measurement. The carrier frequency is set to 2.480 GHz. The ray-tracing technique [21] is used to generate the propagation channel characteristics considering the effect of difference in receive levels and phases between the terminals. To simplify the calculation, we assume a 2-dimentional model of indoor environment neglecting the effect of the floor and the ceiling. We take into account up to 6 times reflections by the walls. The key length is set to 1024 bit. We perform information reconciliation only by using Cascade or the proposed protocol without other information reconciliation protocols such as the

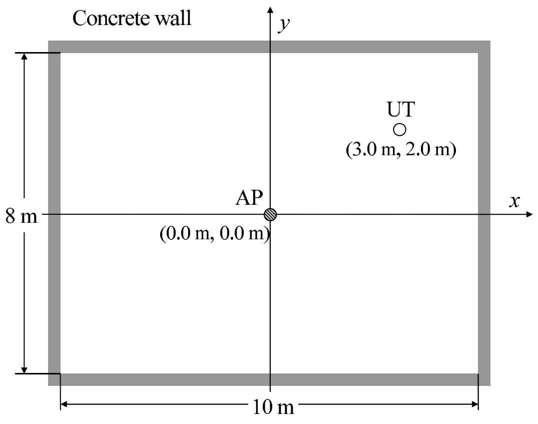


Fig. 5. Simulation environment

Table 1. Parameters of the simulations

Room size	8 m $\times$ 10 m
Positions	AP: (0.0 m, 0.0 m) UT: (3.0 m, 2.0 m)
Antennas	AP: 7-elements ESPAR antenna UT: omni-directional antenna
Carrier frequency	2.480 GHz
Channel model	2D Ray-tracing Reflection: up to 6 times Vertical polarization
Wall material	Concrete ($\varepsilon_r = 6.76$, $\sigma = 0.0023\,\mathrm{S/m}$, $\mu_r = 1$)
Reactance vector	Random
Key length	1024 bit
Information reconciliation	Cascade or proposed protocol

reconciliation protocol using error correcting codes. In Cascade and the proposed protocol, we assume the bit error rate of keys are known, and hence the block length of each pass is optimized according to Eqs. (1) $\sim$ (3).

6 Simulation Results

In this section, we show the comparisons of performance between the proposed protocol and Cascade when SNR varies.

6.1 The Number of Disclosed Bits

Figure 6 shows the number of disclosed bits in information reconciliation until the discrepancies of the legitimate users' keys are completely corrected. In this figure, we also indicate the theoretical limit calculated from Eqs. (4) and (5). From this figure, we observe that by using the proposed protocol, we can decrease the number of disclosed bits at all values of SNR and can achieve performance being close to the theoretical limit of soft-decision information reconciliation.

6.2 The Number of Communications

We consider the number of communications in information reconciliation. We defined the number of communications as the number of times a protocol discloses parities or key bits until the discrepancies of the legitimate users' keys are completely corrected. Figure 7 shows the number of communications in information reconciliation. It can be seen from the figure that when SNR is more than 10 dB, the proposed protocol can correct the discrepancies with less the number of communications than that of Cascade. The reason why the number of communications around SNR of 0 dB is fewer is because the block length around SNR of 0 dB is so short (probably 2 bits) that the discrepancies of the keys can be corrected with a few number of communications.

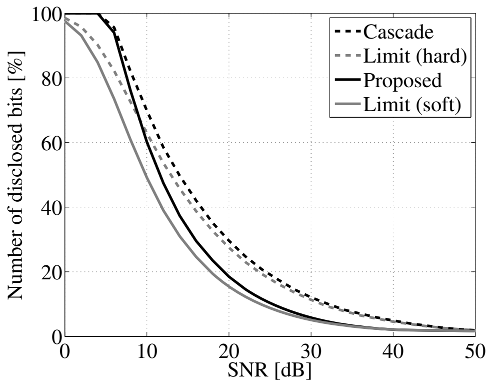


Fig. 6. Number of disclosed bits in information reconciliation

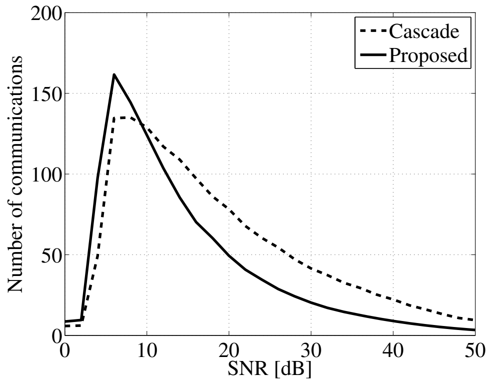


Fig. 7. Number of communications in information reconciliation

6.3 The Efficiency of Protocol

We consider the efficiency of a protocol [15] which indicate how close the protocol is to the theoretical limit. The efficiency of Cascade is defined as follows:

$$\zeta^{\mathrm{hard}} = \frac{H(K_x) - n^{-1}C^{\mathrm{hard}}}{H(K_x) - H(K_x|K_y)} \leq 1, \tag{7}$$

where C^{hard} denotes the number of disclosed bits in Cascade. On the other hand, the efficiency of the proposed protocol is defined as follows:

$$\zeta^{\mathrm{soft}} = \frac{H(K_x) - n^{-1}C^{\mathrm{soft}}}{H(K_x) - H(K_x|R_y)} \leq 1, \tag{8}$$

where C^{soft} denotes the number of disclosed bits in the proposed protocol. Note that the denominators of Eqs. (7) and (8) are different because Cascade is

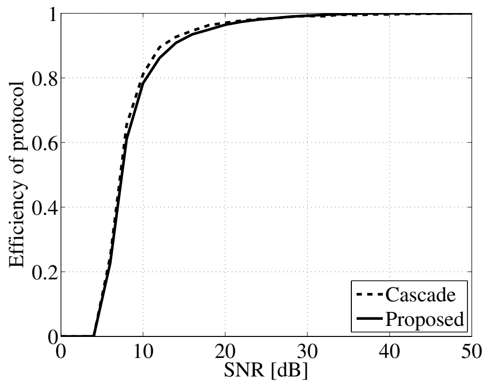


Fig. 8. Efficiency of protocol

hard-decision reconciliation protocol, while the proposed protocol is soft-decision reconciliation protocol.

Figure 8 shows the efficiency of Cascade and the proposed protocol calculated by Eqs. (7) and (8). The efficiency of the proposed protocol is almost identical with that of Cascade. In SNRs above 20 dB, the both achieve efficiency over 0.95. From this result, we can say that the proposed protocol successfully utilizes RSSI information.

7 Conclusion

In this paper, we propose a new information reconciliation protocol using the reliability of the keys for the secret key agreement scheme using the ESPAR antenna. The proposed information reconciliation protocol is a modified version of Cascade protocol utilizing RSSI as the reliability, which is suitable for the secret key agreement scheme using the ESPAR antenna. As the results of the simulations, we can conclude that the proposed protocol can correct errors with less the number of disclosed bits and less the number of communications than those of Cascade.

References

1. Daemen, V.R.J.: The Design of Rijndael: AES–The Advanced Encryption Standard. Springer, Heidelberg (2002)
2. Shannon, C.E.: Communication theory of secrecy systems. Bell syst. Tech. J. 28, 656–715 (1949)
3. Wyner, A.D.: The wire-tap channel. Bell syst. Tech. J. 54(8), 1355–1387 (1975)
4. Csiszár, I., Körner, J.: Broadcast channels with confidential messages. IEEE Trans. Inform. Theory 24(3), 339–348 (1978)
5. Maurer, U.M.: Secret key agreement by public discussion from common information. IEEE Trans. Inform. Theory 39(3), 733–742 (1993)

6. Ahlswede, R., Csiszár, I.: Common randomness in information theory and cryptography–Part I: Secret sharing. IEEE Trans. Inform. Theory 39(4), 1121–1132 (1993)
7. Bennett, C.H., Bessette, F., Brassard, G., Salvail, L., Smolin, J.: Experimental quantum cryptography. J. Cryptology 5(1), 3–28 (1992)
8. Hershey, J.E., Hassan, A.A., Yarlagadda, R.: Unconventional cryptographic keying variable management. IEEE Trans. Commun. 43(1), 3–6 (1995)
9. Aono, T., Higuchi, K., Ohira, T., Komiyama, B., Sasaoka, H.: Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. IEEE Trans. Antennas and Propagation 53(11), 3776–3784 (2005)
10. Wilson, R., Tse, D., Scholtz, R.: Channel identification: secret sharing using reciprocity in UWB channels. IEEE Trans. Inform. Forensics and Security 2(3), 364–375 (2007)
11. Bloch, M., Barros, J., Rodrigues, M., McLaughlin, S.: Wireless information-theoretic security. IEEE Trans. Inform. Forensics and Security 54(6), 2515–2534 (2008)
12. Kawakami, H., Ohira, T.: Electrically steerable passive array radiator (ESPAR) antennas. IEEE Antennas and Propagation Magazine 47(2), 43–50 (2005)
13. Brassard, G., Salvail, L.: Secret key reconciliation by public discussion. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 410–423. Springer, Heidelberg (1994)
14. Buttler, W.T., Lamoreaux, S.K., Torgerson, J.R., Nickel, G.H., Donahue, C.H., Peterson, C.G.: Fast, efficient error reconciliation for quantum cryptography. Phys. Rev. A. 67(5), 052303 (2003)
15. Bloch, M., Thangaraj, A., McLaughlin, S., Merolla, J.: LDPC-based gaussian key reconciliation. In: 2006 IEEE Inform. Theory Workshop (ITW 2006), Punta del Este, Uruguay, pp. 116–120 (2006)
16. Imai, H., Kobara, K., Morozov, K.: On the possibility of key agreement using variable directional antenna. In: 1st Joint Workshop on Inform. Security (JWIS 2006), Seoul, Korea, pp. 153–167 (2006)
17. Bennett, C., Brassard, G., Crépeau, C., Maurer, U.M.: Generalized privacy amplification. IEEE Trans. Inform. Theory 41(6), 1915–1923 (1995)
18. Sugimoto, T., Yamazaki, K.: A study on secret key reconciliation protocol "Cascade". IEICE Trans. Fundamentals E83-A(10), 1987–1991 (2000)
19. Slepian, D., Wolf, J.: Noiseless coding of correlated information sources. IEEE Trans. Inform. Theory 19(4), 471–480 (1973)
20. ZigBee Alliance, http://www.zigbee.org/
21. Goldsmith, A.: Wireless communications. Cambridge University Press, Cambridge (2005)

Protecting Privacy and Securing the Gathering of Location Proofs – The Secure Location Verification Proof Gathering Protocol

Michelle Graham and David Gray

School of Computing, Dublin City University, Dublin, Republic of Ireland
{mgraham,dgray}@computing.dcu.ie

Abstract. As wireless networks become increasingly ubiquitous, the demand for a method of locating a device has increased dramatically. Location Based Services are now commonplace but there are few methods of verifying or guaranteeing a location provided by a user without some specialised hardware, especially in larger scale networks. We propose a system for the verification of location claims, using proof gathered from neighbouring devices. In this paper we introduce a protocol to protect this proof gathering process, protecting the privacy of all involved parties and securing it from intruders and malicious claiming devices. We present the protocol in stages, extending the security of this protocol to allow for flexibility within its application. The Secure Location Verification Proof Gathering Protocol (SLVPGP) has been designed to function within the area of Vehicular Networks, although its application could be extended to any device with wireless & cryptographic capabilities.

1 Introduction

Frequently in applications, one device is required to authenticate itself to another in some way. This interaction often takes the form of exchanging a private secret shared by the two, such as a password, to prove the device's identity. However with ubiquitous networks increasing in prevalence, the issue of context has become more important. Instead of a device being required to produce a password in order to gain access to some resource within the network, it is simply required to prove that its current location is in the area of that network. This *location verification problem* can be solved by technologies such as Active Badge [1] and RFID [2], however these are based within a limited area and require an infrastructure of sensors in order to function. This restricts their usability to small scale networks.

In order to address the issue on a larger scale, we have designed a location verification system for use in large scale networks, specifically Vehicular Adhoc Networks (VANETs), requiring no special physical infrastructure to be in place. We have selected VANETs as a suitable environment in which to apply this location verification scheme due to the protocol's requirement of tamper-resistent devices and public key cryptography.

We propose to verify the presence of a specific device in a particular area using a two step process. First, the claiming device distance bounds [3] with multiple neighbouring devices, to prove that it is within range of other devices known to be in the area. These devices then state whether or not the claiming device successfully proved themself to be within range of them. These proofs are supplied by the claiming device to a central verifying device, who combines them to determine whether the claiming device is in the area.

In the rest of this paper we outline a system for the verification of location claims (Section 3). We then put forth a basic protocol for the protection of the distance bounding exchange. We extend the protocol incrementally to protect the exchange from all tampering and fraudulent attempts (Section 4). Finally, we discuss our future work on this issue (Section 7).

2 Related Work

Calculating the location of a device is a broad area of research. One of the major approaches to solving this problem is location determination. This approach attempts to determine the location of a device without any prior information. GPS is the most widely used technology to locate a device. However readings can be forged and GPS signals can be spoofed [4]. An encrypted version of GPS exists, the Precise Positioning Service (PPS), but this technology is currently restricted.

A second approach to the calculation of a device's location is location verification. Rather than attempting to discover the device's location without apriori knowledge, a device claims to be in a certain region or area. The system then checks whether this claim is legitimate. This approach reduces the complexity of the problem by minimising the area in which the device could be. In [5], the authors introduce the concept of Integrity (I) regions to authenticate messages within wireless networks without the use of pre-existing keys. However, the I-region is very minimal, typically only extending to the limits of the user of a device's field of vision, removing its potential for use in larger scale systems.

In [6], Sastry et al proposed another approach to location verification suitable for larger systems. The Echo protocol employs ultrasound and radio frequency time of flight measurements to gauge the upper bound on the distance between 2 devices. However, a device can employ RF to send its response as far as a colluding device, which can then replay the reply back to the verifier node using ultrasound. As RF has a faster propagation time than ultrasound, the attack would not be detected.

In [7], Waters & Felten proposed the Proximity-Providing Protocol, which is based in wireless networks. Similar to the work done by Sastry et al, the PPP uses round trip times of packets to calculate the upper bound on distance between devices. It relies on a much more ubiquitous technology and is therefore far more practically applicable. However, this protocol is still susceptible to the terrorist fraud. A colluding device could participate in the distance bounding aspect of the protocol in place of the prover, shrinking the latency and making the prover appear closer. We believe we have addressed this issue in our approach, discussed further in Section 4. Our approach also extends the concept behind the Proximity-Proving Protocol to remove the need for a trusted device to be used in the Location Manager role.

Many location verification approaches utilise distance bounding, however it is vulnerable to many circumventing attacks which render the results received incorrect. In [8], Clulow et al outline attacks on current distance bounding approaches, illustrating common flaws in existing protocols. In particular, they cite the use of a single nonce within the challenge-response exchange as vulnerable to a guessing attack. This issue is negated in the Secure Location Verification Proof Gathering Protocol (SLVPGP) through multiple iterations of single nonce challenge-response exchanges. Clulow et al also state that requiring participants to employ encryption during the timed phase of an exchange introduces an inaccuracy into the distance calculations. However, due to the approach to distance bounding employed by the SLVPGP, we believe that this issue does not apply.

3 A System for the Secure Verification of Location Claims

3.1 System Model and Assumptions

Before discussing the design of the SLVPGP, we review the environment in which it is employed and outline the assumptions made regarding participating devices.

1. *Location Claim* - A message containing the location at which a device making a claim purports to be
2. *Claimant* - An untrusted device which makes a location claim.
3. *Verifier* - A trusted entity that decides if a location claim is valid.
4. *Proof Provider* - A neighbouring device contacted by a Claimant to distance bound with, in order to provide proof of a location claim.
5. *External anonymity/confidentiality* - If a device is not involved in an SLVPGP run, they cannot learn location or identity information regarding participating devices even if within range of pertinent transmissions.
6. *Complete anonymity/confidentiality* - A device cannot learn identity or location information regarding another device, even if involved in the same SLVPGP run.

A Claimant wishes to prove its location to a Verifier within an untrusted 802.11 based environment [9]. Although the Claimant and Proof Providers are untrusted devices, we assume that they contain tamper-resistant modules in which all cryptographic keys are stored. This prevents devices from sharing keys in an attempt to sabotage the system's security. The Verifier provides a list of devices in the vicinity of the claimed location for use as Proof Providers. The Claimant distance bounds with each of these to ascertain that it is within range of that device. The results of these exchanges are supplied to the Verifier, which makes its decision based on this information.

The aim of our protocol is to secure the proof gathering process and protect it from intruders and malicious parties within the exchange. We define a secure protocol as one which satisfies the following three security properties. First, *anonymity of identity* - a device's identity should not be discoverable by any party other than the Verifier, except where expressly given to it. Second, *confidentiality of location* - the location of a device with a specific identity should not be discovereable by any party other than where expressly given to it. Finally, *authentication of information* - the origin of any given message must be known.

3.2 Threat Model

Honest nodes behave exactly as their roles dictate. They can communicate with the Verifier, along with anyone within radio range. They receive all messages broadcast within their area, but do not act on messages not intended for them. When a node participates in distance bounding, information regarding their location is leaked [10]. Some solutions to the issue of MAC address identification have been proposed [11,12,13], however this is beyond the scope of this research. The SLVPGP does not allow leaked information to be tied to a specific device's identity within the system, by protecting the identities of those involved.

Malicious nodes fall into one of three categories: malicious Claimants, malicious Proof Providers and malicious nodes external to the exchange, i.e. intruders. Malicious nodes cannot occupy the role of the Verifier as this is a trusted, un-compromisable entity. All malicious nodes can eavesdrop any message sent within the system. Malicious nodes may also manipulate received messages and retransmit them, but cannot prevent a node from receiving a specific message.

We consider multiple attacks on the SLVPGP, most of which are attempted by malicious Claimants attempting to influence the result of their location verification. The first is the *guessing attack*. A malicious Claimant could attempt to guess the correct response to a challenge and send its response prior to receiving the challenge nonce. If successful, the malicious Claimant could prove itself to be within range of the Proof Provider. However, this is addressed through the use of multiple challenge-response iterations, reducing the probability of success. Another attack attempted by a malicious Claimant is the *terrorist fraud* [14]. A malicious Claimant has a proxy act as a man-in-the-middle between himself and the verifying device. This allows the malicious Claimant to convince a Proof Provider that he is closer than he truly is. Finally, in a *snooping attack*, an external intruder node may attempt to gain access to location or identification information through eavesdropping. These attacks are thwarted through the use of encryption on messages containing private information. The case of a malicious proof provider lying about the presence of an honest Claimant is dealt with by the Verifier through the use of trustworthiness levels.

3.3 The Role of the Verifier

The SLVPGP relies upon the Verifier to supply suitable Proof Providers for use in during a run of the protocol, and to determine the final verdict on a claim using their proofs. Unlike the Claimant and its Proof Providers, the Verifier is a trusted device outside the system's environment, possessing the identities and public keys of all devices within the entire Location Verification system. The exact functionality of the Verifier lies beyond the scope of this paper, but we outline here the Verifier's basic processes required by the protocol.

The Claimant does not participate in the selection of Proof Providers for use in a location claim. If involved, it would gain the ability to manipulate the process. Instead, the Verifier geographically routes [15] a request seeking Proof Provider volunteers to the area in which the Claimant purports to be. Only devices in the area of the claimed location receive this request, and Proof Providers are only selected from the volunteers responding to this request. Therefore all Proof Providers used within the protocol are in the correct area.

The Verifier's principle task is to assess the possibility of a device's location claim, based on information gathered for it by the Claimant from Proof Providers. However, the reliability of one Proof Provider's verdict is different to that of another, and so each must be weighted according to that device's past behaviour. This process is achieved using the beta probability density function [16]. Therefore a device with a long record of honest behaviour will have a stronger vote than that of a device with an unreliable past.

3.4 The Use of Distance Bounding in the SLVPGP

In the SLVPGP, distance bounding is not used to calculate specific distances between the Claimant and its Proof Providers. Instead, it allows a Proof Provider to confirm that the Claimant is not attempting to execute a terrorist fraud on the system. In order for the Claimant to participate in distance bounding , they must either be in that area or collude with a device that is. The round trip time required for an exchange when the Claimant is perpetrating a terrorist fraud is drastically different to the time required for an honest exchange. It is this time discrepancy that is being tested when distance bounding is performed. In addition to this, employing distance bounding without requiring specific distances between devices, "off the shelf" wireless devices become feasible, allowing the protocol far greater scope for application.

In order for distance bounding to function as a method of detecting a terrorist fraud attack, the Claimant must be required to actively participate in the exchange. The employment of authentication forces the Claimant to receive the message and compose the corresponding reply, rather than employing a proxy. When dealing with authentication during distance bounding, the concept of using any form of encryption is usually dismissed. When using distance bounding to gauge physical distance, the time required to digitally sign something is thought to overpower the time required to send a message on a round trip. However, we require only that there is a distinction between an honest exchange and an exchange involving a proxy device. As the difference between honest and fraudulent round trip times are quite dramatic prior to the inclusion of digital signatures, we believe that the distinction will hold. We intend to investigate this further in the future (see section 7).

4 The Secure Location Verification Proof Gathering Protocol (SLVPGP)

4.1 Protocol Outline

In order to understand what security risks are present in the proof gathering process, we first model a protocol devoid of any security. We outline the protocol as a sequence of steps taken by the parties involved in the exchange: the Verifier (V), the Claimant (C) and N Proof Providers (B_i where $i \in \{1...N\}$).

1. C→V: C, X_C
   The Claimant transmits its identity (C) and location (X_C) to the Verifier, requesting that its location claim be verified.
2. V→C: $B_1, B_2, ...B_N$
   The Verifier sends a list of Proof Providers to the Claimant.
3. C→B_i: $B_i, \mathcal{N}_i, \mathcal{N}_i^{'}$
   The Claimant broadcasts a message for each Proof Provider containing their identity and two nonces (long random integers), $\mathcal{N}_i$and $\mathcal{N}_i^{'}$.
4. The Claimant and Proof Providers create a chain of M hashes for each nonce. These are noted as $H_{i,k}$ and $H_{i,k}^{'}$ respectively, where $i \in \{1...N\}$ and $k \in \{1...M\}$.
5. Distance Bounding
   This stage is performed multiple times to lessen the effect of any network issues and to reduce the effectiveness of a guessing attack.
   (a) B_i starts its timer.
   (b) B_i →C: k, $H_{i,k}$, $\mathcal{N}_i^{''}$
   B_i sends a message to the Claimant containing a new random nonce $\mathcal{N}_i^{''}$, a randomly selected value from the hash chain of $\mathcal{N}_i$, $H_{i,k}$, and its position in the chain, k. The value of k decreases with each distance bounding iteration. The Claimant checks whether the kth value in the $\mathcal{N}_i$ hash chain matches $H_{i,k}$ and if so, continues to the next step.
   (c) C→B_i: $H_{i,k}^{'}$, $\mathcal{N}_i^{''}$
   The Claimant sends a message to B_i containing $\mathcal{N}_i^{''}$and the kth value in the $\mathcal{N}_i^{'}$ hash chain. B_i compares this with the kth value in its $\mathcal{N}_i^{'}$ hash chain. If the two values match and the received value of $\mathcal{N}_i^{''}$ matches its own sent value, B_i stops its timer and calculates the round trip latency (subtracting out its own internal processing time).
6. B_i →C: T_i, X_i, L_i
   B_i sends the Claimant its proof, comprised of its current location (X_i) and overall decision regarding the presence of the Claimant in its vicinity (L_i). A timestamp is also included to tie the proof to this specific point in time.
7. C→V: C, T_C, X_C, $\{T_1, X_1, L_1\}$, ..., $\{T_N, X_N, L_N\}$
   The Claimant compiles all the proofs gathered from its Proof Providers and forwards them to the Verifier, along with its identity C, current location X_C and a timestamp T_C to tie the proofs to that point in time.

4.2 Protocol Discussion

Due to the possibility of network issues such as delays and lost packets, the result of a location claim could be affected through the true results of a single distance bounding exchange being distorted. We have included multiple distance bounding exchanges within the protocol to compensate for this possibility. In order to avoid high data overheads, we employ hashing [17] to calculate multiple nonces from a single initial nonce. In this method, the kth nonce $H_{i,k}$ in a hash chain is calculated from $\mathcal{N}_i$ using the formula $H_{i,k} = \mathcal{H}^k(\mathcal{N}_i)$, i.e. the hash of the initial nonce is hashed and this third value is then hashed to produce a fourth value etc.

The hash chains produced from $\mathcal{N}_i$ and $\mathcal{N}_i^{'}$ allow the Claimant and Proof Provider B_i to validate that they are interacting with their expected device. $\mathcal{N}_i$ proves the message's origin is from a Proof Provider engaged in the Claimant's current location claim. The value from the hash chain derived from $\mathcal{N}_i^{'}$ verifies that the message received originated from the Claimant involved in that specific location claim. $\mathcal{N}_i^{''}$ is used to ensure that the Claimant cannot guess which nonce in the chain to send and transmitting his response before receiving the Proof Provider's message. As the Claimant's reply contains $\mathcal{N}_i^{''}$, it is forced to wait until it receives this value. This dramatically reduces the possibility of an early transmission attack successfully being accepted as a valid response.

The use of a randomly selected hash $(H_{i,k})$ increases the unpredictability of the Proof Provider's distance bounding challenge message for the Claimant. However, given the value $H_{i,k}$, an intruder could then calculate all values above it in the hash chain. For this reason the value of k decreases with each iteration of distance bounding, moving downwards within the chain. Although this reduces the reusability of the hash chain, it removes the risk of a security leak.

The Proof Providers measure the latency of each exchange, and these latencies are used to calculate the Proof Provider's final verdict, a boolean representing either a positive or negative result. However, the protocol as shown above upholds none of the security protocols outlined in section 3.1. Both the distance bounding exchange and the verdicts are susceptible to tampering, either by an intruder or a malicious Claimant. In addition to this, proofs can be fabricated by a malicious Claimant without any interaction with a Proof Provider. Finally, the protocol is vulnerable to the terrorist fraud as the distance bounding messages are not tied to the Claimant. In order to address these vulnerabilities, we have repeatedly extended the basic protocol, increasing security with each extension.

5 Extending the Protocol

As before, we outline these protocols as a series of steps taken by the Claimant (C), Proof Providers (B_i where $i \in \{1...N\}$) and Verifier (V). We assume that all parties involved have asymmetric key pairs associated with them. These key pairs will be noted as K_A^- and K_A^+ for the private and public keys respectively, where A is the owning party's identity.

Algorithm 1. SLVPGP Extension 1

```
1. C→V: $\{—C, X_C—\}_{K_C^-}$
2. V→C: $\{|B_1, B_2, ..., B_N|\}_{K_V^-}$
3. C→ $B_i$: $B_i$,$\{\mathcal{N}_i, \mathcal{N}'_i\}_{K_{B_i}^+}$
4. The Claimant & Proof Providers create a chain of $M$ hashes for each nonce.
5. Distance Bounding (executed multiple times)
   (a) $B_i$ starts its timer.
   (b) $B_i$ →C: $k$, $H_{i,k}$,$\mathcal{N}_i''$
   (c) C→$B_i$: $H'_{i,k}$, $\{|\mathcal{N}_i''|\}_{K_C^-}$
6. $B_i$→C: $\{|T_i, X_i, L_i, C|\}_{K_{B_i}^-}$
7. C→V: $\{|T_C, X_C|\}_{K_C^-}$,$\{|T_1, X_1, L_1, C|\}_{K_{B_1}^-}$ , ..., $\{|T_N, X_N, L_N, C|\}_{K_{B_N}^-}$
```

5.1 The SLVPGP: Extension 1

The protocol extension shown in algorithm 1 removes many of the vulnerabilities outlined regarding the basic protocol. Encrypting $\mathcal{N}_i$ and $\mathcal{N}'_i$ prevents intruding devices from gaining knowledge of these values without collusion. The addition of a digitial signature to the echoing nonce removes the danger of a collusion attack going undetected & provides authentication. A malicious Claimant can no longer fabricate proof for the Verifier, nor can it undetectably alter the content of an existing proof. Similarly, an intruder cannot fraudulently participate in the exchange. Finally, the inclusion of the Claimant's identity within the signed proof message from the Proof Provider prevents malicious Claimants from using valid proofs pertaining to another Claimant as evidence of their location claim.

5.2 The SLVPGP: Extension 2

In algorithm 1, we extended the protocol to provide authentication and prevent many attacks such as the terrorist fraud. In this section we further extend the protocol (algorithm 2) to provide an increased level of anonymity and confidentiality. We achieve this through the addition of encryption.

The addition of encryption builds upon this framework to include both external anonymity and external confidentiality. While the Claimant and all involved Proof Providers have access to the location and identity information being sent over the run of the protocol, no other devices can discover it, whether they are within range of the transmissions or not.

5.3 The SLVPGP: Extension 3

Though the previous extension increases the protocol's level of security, honest participants could still record information on the parties involved, then use this information maliciously at some point in the future. For this reason, we wish to provide complete anonymity and confidentiality (algorithm 3).

Algorithm 2. SLVPGP Extension 2

```
1. C→V: $\{\{—C, X_C—\}_{K_C^-}\}_{K_V^+}$
2. V→C: $\{\{—B_1, B_2, ..., B_N—\}_{K_V^-}\}_{K_C^+}$
3. C→ $B_i$:$\{B_i, C, \mathcal{N}_i, \mathcal{N}'_i\}_{K_{B_i}^+}$
4. The Claimant & Proof Providers create a chain of $M$ hashes for each nonce.
5. Distance Bounding (see notation in [1]).
6. $B_i$ →C: $\{\{|T_i, X_i, L_i, C|\}_{K_{B_i}^-}\}_{K_C^+}$
7. C→V:$\{\{|X_C, T_C|\}_{K_C^-}\}_{K_V^+}$,
   $\{\{|T_1, X_1, L_1, C|\}_{K_{B_1}^-}\}_{K_V^+}$, ..., $\{\{|T_N, X_N, L_N, C|\}_{K_{B_N}^-}\}_{K_V^+}$
```

Algorithm 3. SLVPGP Extension 3

```
1. C→V: $\{\{—C, X_C—\}_{K_C^-}\}_{K_V^+}$
2. V→C: $\{— \{B_1, \mathcal{N}_1, \mathcal{N}'_1, \mathcal{N}'''_1\}_{K_{B_1}^+}, \{\mathcal{N}_1, \mathcal{N}'_1, \mathcal{N}'''_1\}_{K_C^+}$, ...,
   $\{B_N, \mathcal{N}_N, \mathcal{N}'_N, \mathcal{N}'''_N\}_{K_{B_N}^+}, \{\mathcal{N}_N, \mathcal{N}'_N, \mathcal{N}'''_N\}_{K_C^+} —\}_{K_V^-}$
3. The Claimant decrypts and stores each nonce received from the Verifier.
4. C→ $B_i$: $\{B_i, \mathcal{N}_i, \mathcal{N}'_i, \mathcal{N}'''_i\}_{K_{B_i}^+}, \{C\}_{K_V^+}$
5. The Claimant and Proof Providers create a chain of $M$ hashes for each nonce.
6. Distance Bounding (see notation in [1]).
7. $B_i$ →C: $\mathcal{N}'''_i, \left\{|\{|\mathcal{N}''_i|\}_{K_C^-}, \mathcal{N}''_i|\right\}_{K_{B_i}^-}, \{\{|T_i, L_i, X_i, \{C\}_{K_V^+}|\}_{K_{B_i}^-}\}_{K_V^+}$
8. C→V: $\{\{|X_C, T_C|\}_{K_C^-}\}_{K_V^+}$,
   $\left\{|\{|\mathcal{N}''_1|\}_{K_C^-}, \mathcal{N}''_1|\right\}_{K_{B_1}^-}, \{\{|T_1, L_1, X_1, \{C\}_{K_V^+}|\}_{K_{B_1}^-}\}_{K_V^+}$,
   ..., $\left\{|\{|\mathcal{N}''_N|\}_{K_C^-}, \mathcal{N}''_N|\right\}_{K_{B_N}^-}, \{\{|T_N, L_N, X_N, \{C\}_{K_V^+}|\}_{K_{B_N}^-}\}_{K_V^+}$
```

However, the protocol is now vulnerable to a *denial of service attack*, where the Claimant is supplied with fraudulent proofs. This is because the Claimant is unable to differentiate between valid and fraudulent proof messages, as they are now encrypted with an unknown key. In order to solve this issue, the Claimant requires a method of matching a received proof with an unknown Proof Provider. $\mathcal{N}'''_i$ is used to verify that all proof messages received during an exchange were created by legitimate Proof Providers, removing an intruder's ability to undetectably insert illegitimate proofs into the exchange.

The Verifier encrypts the message containing the nonces, to prevent intruders from learning their values. However, as the Claimant also requires a copy of these values, the Verifier is forced to create and encrypt two messages. Broadcasting allows the Claimant to communicate with its Proof Providers without knowledge of their identities and vice versa, allowing complete anonymity and confidentiality. Therefore, we believe we have designed a secure protocol, satisfying our outlined security properties.

6 Overall Analysis

6.1 Cost Analysis

Each extension to the SLVPGP increases the security provided to those participating. However, these improvements in security also require extra data to be transmitted and extra time to be taken to compute encryptions or digital signatures. We define the *costs* of an extension to mean this extra data and/or time required.

The costs related to the first extension stem from the addition of digital signatures to protocol messages. As signatures must be calculated and transmitted, this generates both time and data transmission costs. However, these costs are minimal in relation to those incurred by the second and third extensions. The addition of encryption to these levels increases both the data and time costs required. The third extension is the most costly, however, due to the creation and transmission of duplicate messages to allow for complete anonymity and confidentiality.

Each of the three extensions to the SLVPGP are fully secure against the terrorist fraud, in addition to maintaining the integrity of the proof messages supplied to the Verifier by the Claimant. With the exponential increase in data overheads incurred within the third extension, complete anonymity and confidentiality come at quite a high cost. The second extension to the SLVPGP is the most widely applicable, providing anonymity and confidentiality against all devices external to an exchange without incurring extreme costs.

6.2 Security Analysis

We believe the protocol presented above is secure and satisfies the outlined security properties. Formal verification of the protocol hierarchy has been carried out to confirm this using the model checker FDR [18], along with Casper [19] to automatically generate CSP [20] descriptions. When writing in a language such as CSP, the density and complexity of the code often result in overlooked errors and logic flaws. Using Casper allows us to automate the production of the mathematical model description of the protocol, reducing the liklihood of errors occuring.

A script was created to model each extension, stipulating only the individual security properties claimed above. After each extension's script was checked and passed, the intruder's knowledge was amended to include the secure keys of individual participants. This allowed for the modelling of malicious participants. Even with security intentionally compromised in this manner, the security properties for each extension are upheld, excluding those related to the compromised device. This confirms that the intended security properties are supported and the protocol does not appear to have any weaknesses vulnerable to attack by intruders.

7 Future Work

We intend to analyse the effect of malicious and colluding Proof Providers on the system through simulations of the Verifier's calculations using sample proofs. In addition to this, we intend to analyse the number of proof providers required by the system for a proof to be successful, in both honest and hostile environments.

Another area of focus is the simulation of distance bounding including digital signatures. At present, our system assumes that distance bounding is capable of detecting a terrorist fraud through differenciating between a reasonable time for an honest exchange and the time required for a proxy exchange to complete. However, we intend to confirm this assumption through the simulation of both honest and proxy distance bounding exchanges encorporating digital signatures.

8 Conclusion

In this paper, we discussed the demand for a method of locating a device within a wireless network. We then focused on the problem of location verification and outlined our system for verifying a device's location claim. This allows a Claimant to prove its location to a central verifying agent through gathering proof of its proximity to local devices. Unlike previously existing approaches however, the system does not require trusted agents to act as Proof Providers. Instead, we make use of unknown and untrusted devices local to the Claimant, in possession of a tamper-resistant module, to protect all cryptographic keys. As the system is designed for use in an untrusted environment and the participants in the exchange are also untrusted, we noted that a Claimant merely distance bounding with local devices would not provide valid proofs for use within the system and that a method of protecting the integrity of information passed within it is needed. In order to secure this process from attack by both intruders and malicious participating devices, we have presented the Secure Location Verification Proof Gathering Protocol. We repeatedly extended this protocol to provide increasing levels of security, culminating in the provision of authentication, complete anonymity and confidentiality within the final protocol. This incremental approach to the protocol's design allows for flexibility in terms of cost vs benefit.

References

1. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. 10(1), 91–102 (1992)
2. Sarma, S.E., Weis, S.A., Engels, D.W.: RFID systems and security and privacy implications. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 454–469. Springer, Heidelberg (2003)
3. Brands, S., Chaum, D.: Distance-bounding protocols (extended abstract). In: Theory and Application of Cryptographic Techniques, pp. 344–359 (1993)
4. Gabber, E., Wool, A.: How to prove where you are: racking the location of customer equipment. In: CCS 1998: Proceedings of the 5th ACM conference on Computer and communications security, pp. 142–149. ACM, New York (1998)

5. Čapkun, S., Čagalj, M.: Integrity regions: authentication through presence in wireless networks. In: WiSe 2006: Proceedings of the 5th ACM workshop on Wireless security, pp. 1–10. ACM, New York (2006)
6. Sastry, N., Shankar, U., Wagner, D.: Secure verification of location claims. Technical report (2003)
7. Waters, B., Felten, E.: Secure, private proofs of location. Technical report, Princeton University (2003)
8. Clulow, J., Hancke, G.P., Kuhn, M.G., Moore, T.: So near and yet so far: Distance-bounding attacks in wireless networks. In: Security and Privacy in Ad-Hoc and Sensor Networks, pp. 83–97. Springer, Heidelberg (2006)
9. Association, I.S.: Ieee standards for information technology – specific requirements – part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Published online (1999)
10. Rasmussen, K.B., Čapkun, S.: Location privacy of distance bounding protocols. In: CCS 2008: Proceedings of the 15th ACM conference on Computer and communications security, pp. 149–160. ACM Press, New York (2008)
11. Lindqvist, J., Takkinen, L.: Privacy management for secure mobility. In: WPES 2006: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 63–69. ACM, New York (2006)
12. Huang, L., Matsuura, K., Yamane, H., Sezaki, K.: Enhancing wireless location privacy using silent period. In: 2005 IEEE Wireless Communications and Networking Conference, pp. 1187–1192 (2005)
13. Gruteser, M., Grunwald, D.: Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. Mobile Networks and Applications 10, 315–325 (2005)
14. Desmedt, Y.: Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In: Proceedings of SECURICOM 1988, Sixth Worldwide Congress on Computer and Communications Security and Protection, pp. 147–159 (1988)
15. Mauve, M., Widmer, A., Hartenstein, H.: A survey on position-based routing in mobile ad hoc networks. IEEE Network 15, 30–39 (2001)
16. Josang, A., Ismail, R.: The beta reputation system. In: e-Reality: Constructing the Economy (2002)
17. Haller, N.M.: The S/KEY one-time password system. In: Proceedings of the Symposium on Network and Distributed System Security, pp. 151–157 (1994)
18. Roscoe, A.W.: Modelling and verifying key-exchange protocols using csp and fdr. In: Computer Security Foundations Workshop, p. 98 (1995)
19. Lowe, G.: Casper: A compiler for the analysis of security protocols. In: Computer Security Foundations Workshop IEEE, p. 18 (1997)
20. Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. ACM 31, 560–599 (1984)

Providing Strong Security and High Privacy in Low-Cost RFID Networks

Mathieu David and Neeli R. Prasad

Center for TeleInFrastruktur (CTIF), Aalborg University,
Niels Jernes Vej 12, 9220 Aalborg Øst, Denmark
{md,np}@es.aau.dk

Abstract. Since the dissemination of Radio Frequency IDentification (RFID) tags is getting larger and larger, the requirement for strong security and privacy is also increasing. Low-cost and ultra-low-cost tags are being implemented on everyday products, and their limited resources constraints the security algorithms to be designed especially for those tags. In this paper, a complete solution providing strong security and high privacy during the whole product lifetime is presented. Combining bit-wise operations and secret keys, the algorithm proposed addresses and solves all the common security attacks.

Keywords: RFID tags, security, bit-wise, privacy.

1 Introduction

In wireless communications, the security aspect has always been a big issue. Maintaining the confidentiality and the integrity of the data is a main concern, as people expect to send information to a targeted authority only. Designing a system using wireless communications means considering a large number of security threats. Those threats will be listed further in this paper.

1.1 The RFID Technology

(RFID technology is not recent. It has been used for the first time during World War II with the "Identification Friend or Foe" (IFF) system to make the distinction between allied and enemies' planes. It has been used afterwards for many years in very specific areas to perform particular tasks such as automatic car payment on the highway or cattle monitoring. Its relatively expensive price at that time made it difficult to be generalized. Thanks to efforts of miniaturization and improvement in the technology, it became possible to produce RFID Tags and readers for a relatively low price, leading to a renewed interest in this technology.

The RFID Technology consists of different devices that communicate over radio transmissions. A typical RFID network is composed of RFID Tags, a reader and a database (see Figure 1). The reader is able to communicate both with the database and the tags, while the other devices can only communicate with the reader. To retrieve information from the RFID Tag, the reader sends a request. The Tag (which is most often a passive device) uses the energy harvested from the request to send back an

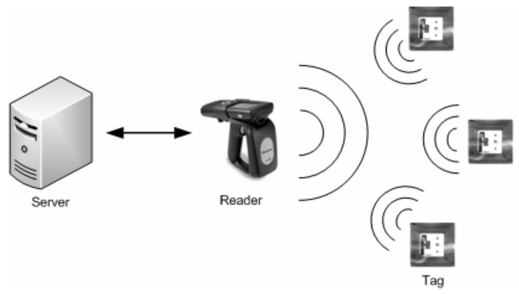


Fig. 1. RFID Architecture

answer. The aim of the RFID technology is to replace the barcode, introducing the Internet of things, where every single product has its own code.

1.2 A New Problematic

With the development of the RFID technology, a new kind of threat appears; the privacy threat. Privacy has become one of the most sensitive topics, since it has to deal with not only the privacy and integrity of the company, but also with the privacy of the consumer, who is the end-user in the production chain. These privacy threats become a reality as the deployment of RFID tags on everyday products is propagating, but even more because the RFID readers are small, relatively inexpensive, and have sufficient processing capabilities to read most of the tags. Solving those threats is one of the most challenging topics regarding this technology, both for the research aspect as for the social aspect it introduces.

1.3 Privacy Threats

So far, seven main privacy threats have been identified, compromising the privacy of the consumer:

The action threat concerns the behaviour or intent of a user that can be inferred from the evolution of the group of tags surrounding him.

The association threat focuses on the product itself. Not only the kind of products the person owns, but the precise product can be discovered (very limited edition, for example).

The location threat deals with tracking of people thanks to the tags they are wearing. Since most of the readers are fixed, it can be quite easy to monitor someone's location through the whole day, by checking all the places where some of the tags he is wearing, have been read.

The preference threat is related to the specific kind of product someone owns and buys, to define his consumer profile, and thus target him more specifically (target advertising, for example).

The constellation threat is highly related to the location threat except that it is not a targeted individual that is tracked but a random individual without knowing its identity.

The tracking of this person would be performed by tracking the constellation of RFID tags that he is wearing.

In **the transaction threat**, the tracking of goods does not stop at the consumer step, but goes further and keep on tracking the location and ownership of the tagged object through its entire product life (until the chip is destroyed). If some product goes from one person to another, you can conclude that they know each other and draw a link between them. Step by step, you can draw a complete social network, connecting all the links between people.

The breadcrumb threat is the issue that links someone to the objects he bought as long as the objects exist (i.e. the tag is working). When someone buys a product in a retail shop, the tag information is stored in the shop database (or even a larger database), and is not updated after the consumer's purchase. It can then create some trouble to the owner in case of a misuse by a third person.

We can infer from all those threats that the consumer privacy can be highly exposed if no action is done to avoid them. They are described in more details in [1]. Many researchers have been working on this topic and their contribution to this field is detailed in the next section.

2 Related Works

While facing an issue, two options are available. One consists in finding the solutions to solve the issue. In this particular case, the solution is to implement encryption and authentication in the process. Some research has been done to compare all the available encryption algorithms for ultra-low power devices [2]. However, this work is more focused on Wireless Sensor Networks, which embedded more computational resources than RFID Tags. A similar survey is presented in [3] and goes even more deeply in the energy consumption of each single computational operation. Many different ways have been explored to ensure security in RFID technology. Some works focus on the physical properties of the device to maintain security, using the physical imperfections of the hardware to guarantee authenticity [4]. Several security algorithms have been proposed as well, some relatively energy-consuming introducing "lightweight" elliptic curve cryptography primitives [5] or trapdoor-based mutual authentication [6]. Others are using considerable memory resources either through a key-table [7] or storing additional data to preserve untraceability [8]. Some propose a secured solution limited to a single reader scenario [7][9], which could be an unrealistic constraint in real case deployment. Finally, a few other protocols perform strong security and authenticity with simple bit-wise operations [10], [11]. These two last mentioned protocols seem to be the best alternative for low-cost RFID solutions, since they are not as energy-consuming and memory-demanding as the others, and do not present some major constraints or security gaps.

The other option consists in removing the issue itself. Authentication and encryption is a strong requirement in a wireless communication process to maintain privacy. However, this requirement becomes useless without communication. In fact, all the privacy threats are based on a simple assumption: tags and readers are able to communicate. As communications stop, the threats disappear (except the breadcrumb threat which is related to the physical product itself). So, the concept is to avoid communication between tags and reader. In this way, a possible solution was the use of a battery-powered mobile device called "RFID Guardian", supposed to create interference around the guardian to preserve the privacy of the RFID Guardian holder [12]. The same principle has been studied by the RFID expert A. Juels et al. who propose a "blocker tag" that selectively allow communications with authenticated readers [13]. Those solutions present the disadvantage to constrain the user to hold an additional device permanently with him, and open the path to security gaps in case the device fails. The ultimate alternative is to temporarily deactivate the tag to make sure it is not able to communicate anymore, without purpose.

3 Proposed Solution

3.1 A Few Assumptions

In the production cycle, the privacy concerns appear at the very last link of the chain, when the consumer buys the product. The concept of the proposed solution is to ensure a strong security during the whole production process (see Fig. 2), and a strong privacy, once the consumer owns the product.

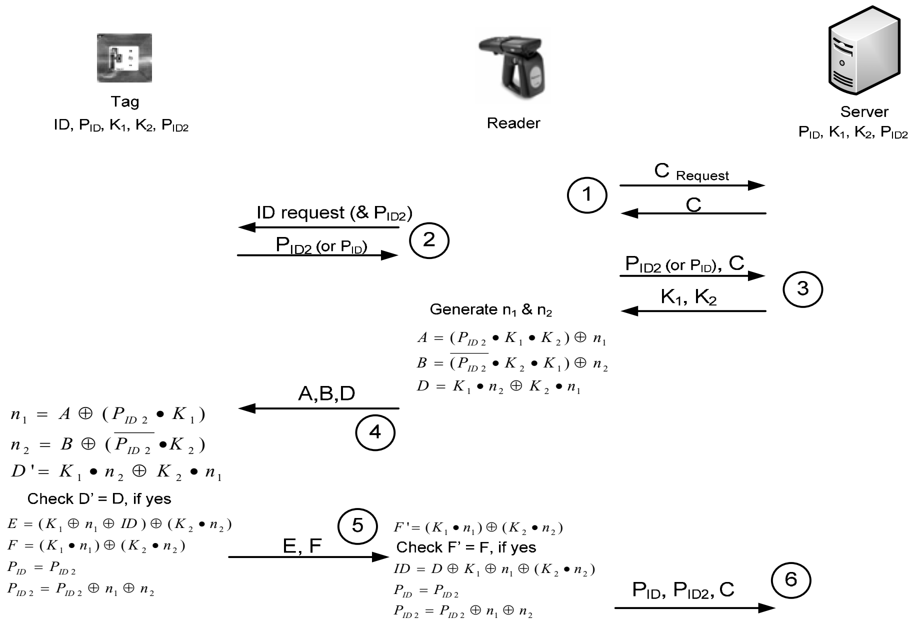


Fig. 2. Security Protocol

The security protocol is inspired by the works done in [10] and [11] combining a strong authenticity and reducing the computational load of the tag, without compromising the security. While the tag is created, four numbers are shared by the tag and the server; two Pseudo ID (P_{ID} & P_{ID2}, initially equals) and two keys, K_1 and K_2. All of them have a length equal to the tag ID. In this protocol, we assume that the link between the Reader and the Server, which are two powerful devices, is secured against all the traditional attacks related to networks. We also assume that the reader will always be able to communicate with the server, since readers are often equipped with the GSM technology, making the communication available everywhere, anytime.

3.2 The Protocol

- **1st step:** The reader requests a certificate of authenticity to the server. The server decides whether or not the reader is allowed to retrieve further information from the server, by sending back a certificate. This first step is done only once per reader, per day. If the reader wants to read multiple tags at the same time, it will just request a certificate once, and use it for all the tags. If the reader already has a valid certificate (the one of the day) the protocol starts in Step 2.
- **2nd step:** The reader sends a request to the tag, which replies with its P_{ID2}. If the reader cannot find a match in the database, it will send another request with the P_{ID2} to the tag, which will reply with its P_{ID}.
- **3rd step:** The reader sends the P_{ID} together with its certificate. If the certificate is authentic, the server replies with K_1 and K_2. The reader then generates 2 random numbers n_1 and n_2. It computes A, B and D with the keys shared by the tag and the random numbers.
- **4th step:** The tag computes the values of n_1, n_2 and D'. If K_1 and K_2 are genuine, D' and D will be equal. This step is necessary to authenticate the reader from the tag's point of view. Since K_1 and K_2 are not exchanged between tag and reader, a match in the D value means that the reader is legitimate.
- **5th step:** The tag computes E with its ID, F with the 4 secret values, and sends them to the reader. The reader will first check the value of F to check the authenticity of the tag. If the values match, it will be able to retrieve the ID from E using K_1, K_2, n_1 and n_2. Both reader and tag compute a new value of Pseudo ID (P_{ID2}) that will be used for the next communication.
- **6th step:** The reader sends an updated version of the Pseudo ID (P_{ID2}) as well as the previous version (P_{ID}) and its certificate, to maintain authenticity.

Thanks to n_1 and n_2 the values A, B, D, E and F are always different. Changing the value of P_{ID} is a necessity to avoid tracking of the tag. In fact, it's rather easy to track a tag that would always reply with the same message to a simple request. Storing the previous value of the P_{ID}, while the next value to be used is P_{ID2}, is to maintain the integrity of the network and avoid de-synchronization attacks.

4 Security Evaluation

4.1 Security Analysis

In this section, we will review a bit more in detail the security threat and the solutions proposed by the algorithm.

Eavesdropping happens when an attacker listens to the channel to retrieve information. Even if the attacker will receive the messages, it is not possible to determine the values of the secrets keys or the ID of the tag, since the messages are encrypted (A, B, D, E and F).

Relay attack occurs when a fake tag and a fake reader try to counterfeit a legitimate authentication. It cannot be done in the proposed solution because tag and reader don't exchange data to authenticate each others directly. The reader authenticates itself with the server, the tag authenticates the reader through D and the reader authenticates the tag through F. Any minor change in those values will be detected.

Unauthorized tag reading and **tag cloning** are solved through the use of authentication. If the reader is not allowed by the server it will never get any information related to the tag. Similarly, if the tag is not genuine, it will not be able to decrypt the values of A, B and D.

Tracking is done by simply listening to the data transmitted by the tag. Since the data sent by the tag is always different, P_{ID2} is changing in a random way since it involves n_1 and n_2, it is not possible to track the tags over time. Though, between two successful authentications, a malicious reader will always receive either P_{ID} or P_{ID2} from its requests and will therefore be able to track the tag. However, it assumes that the malicious reader is following the tag, which does not make sense in a realistic scenario (i.e. if the reader follow the tag, you do not need to read it in order to track it).

Replay attack occurs when a malicious tag tries to authenticate itself by repeating the authentication sequence (P_{ID}) of a genuine tag. While the reader will reply with A, B and D, the malicious tag won't be able to retrieve any information from those values.

De-synchronization attack is used by an attacker to update the values in only one part of the network, either the tag or the reader, in order to make it impossible for them to communicate further. In our scheme, before any update of data, there is a check (D and F) and if the values do not match, the intrusion is detected and the value of P_{ID2} is not updated in the server. If it is updated by mistake, the old value (P_{ID}) will be used to recover from the attack.

Forward Security is the possibility to maintain integrity of the communication over time. It means that even if the tag is physically compromised one day, and the attacker is able to recover the secret values of the tag, it will not be able to find the previous data, since every exchange of data includes two random numbers.

Disclosure attack is used to retrieve some secret information from one entity by sending a slightly modified message to see the impact on the answer. In our scheme, any change is detected and the attacker won't receive any answer.

As a conclusion of the security analysis, it appears that the proposed protocol is robust to any kind of attack. The Table 1 is a comparison of security threat in different ultra-lightweight protocols implemented for low-cost RFID tags. Our solution is slightly lighter than SASI and the authentication protocol is more advanced.

4.2 Privacy Analysis

We will review in this section the privacy issues presented in the section I and see how the protocol handle with them. The **action threat** as well as the **association threat** is solved since it is impossible to retrieve information from the tag without being authenticated by the server. The **preference threat**, the **transaction threat** and the **breadcrumb threat** are not directly in the scope of this paper since they deal with the association of a tag to its owner. This is mainly dependant on the application used

Table 1. Comparison of ultra-lightweight authentication protocols

	LMAP	**M²AP**	**EMAP**	**SASI**	**Ours**
Resistance to de-synchronization attacks	No	No	No	Yes	Yes
Resistance to disclosure attacks	No	No	No	Yes	Yes
Privacy and anonymity	No	No	No	Yes	Yes
Forward Secrecy	No	No	No	Yes	Yes
Mutual Authentication	No	No	No	Partially*	Yes
Memory size on tag	6L**	6L	6L	7L	5L
Memory size on server	6L	6L	6L	4L	4L
Operation types on tag	$\oplus, \bullet, +, 2^m$	$\oplus, \bullet, +, 2^m$	$\oplus, \bullet, +, 2^m$	$\oplus, \bullet, +, 2^m, Rot$	$\oplus, \bullet$

* Authentication Reader/server is assumed, Tag/Reader is done only in one direction.
** L denotes the bit length of one pseudonym or one key.
2^m denotes the addition modulo 2.

and the usage of the tag. However, assuming the identity of someone would be revealed, it still remains impossible for an adversary to read the information stored in the tag, insuring the privacy of the person. Finally, the **location threat** and the **constellation threat** are probably the weakest link in this protocol. Both are related to tracking and we saw in the security analysis that between two legitimate readings, it is possible to track a tag. However, it assumes that it exist several malicious readers, placed in different strategic points, able to communicate with each others, and that no legitimate reading is done between the two malicious readings. The nature of the threat depends on the reader's ability to retrieve the identity of the tag's holder.

5 Conclusion

In this paper we presented a new solution to the security and privacy issue in RFID Technology. The solution we provided maintains a strong security despite its ultra-lightweight algorithm, and can overcome any kind of attack over the radio. The comparison with other similar protocols shows that it is more light, simple and at least as robust as the others. Thanks to its light weight and bit-wise operations, it can be easily implemented on ultra-low-cost RFID tags.

Acknowledgements. This work was supported in part by the European Commission, in the context of the project ASPIRE "Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications", grant agreement FP7 ICT-2007-215417.

References

1. Garfinkel, S.L., Juels, A., Pappu, R.: RFID privacy: An overview of problems and proposed solutions. IEEE Security & Privacy 3(3), 34–43 (2005)
2. Kaps, J.-P., Sunar, B.: Cryptography on a Speck of Dust. Computer 40(2), 38–44 (2007)
3. Eisenbarth, T., Kumar, S.: A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test of Computers 24(6), 522–533 (2007)
4. Bolotnyy, L., Robins, G.: Physically Unclonable Function-Based Security and Privacy in RFID Systems. In: The Fifth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2007, March 19-23, pp. 211–220 (2007)
5. SungJin, K., YoungSoo, K., SeokCheon, P.: RFID Security Protocol by Lightweight ECC Algorithm. In: The Sixth International Conference on Advanced Language Processing and Web Information Technology, ALPIT 2007, August 22-24, pp. 323–328 (2007)
6. Hwaseong, L., Eun Young, C., Su-Mi, L., Dong Hoon, L.: Trapdoor-based Mutual Authentication Scheme without Cryptographic Primitives in RFID Tags. In: The Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007, July 19, pp. 73–78 (2007)
7. Huafei, Z., Bao, F.: Securing RFID Tags: Authentication Protocols with Completeness, Soundness, and Non-Traceability. In: The IEEE Wireless Communications and Networking Conference, WCNC 2007, March 11-15, pp. 2698–2702 (2007)
8. Shucheng, Y., Kui, R., Wenjing, L.: A Privacy-preserving Lightweight Authentication Protocol for Low-Cost RFID Tags. In: The IEEE Military Communications Conference, MILCOM 2007, October 29-31, pp. 1–7 (2007)
9. Yung-Chin, C., Wei-Lin, W., Min-Shiang, H.: RFID Authentication Protocol for Anti-Counterfeiting and Privacy Protection. In: The 9th International Conference on Advanced Communication Technology, Feburary 12-14, vol. 1, pp. 255–259 (2007)
10. Il Jung, K., Eun Young, C., Dong Hoon, L.: Secure Mobile RFID system against privacy and security problems. In: The Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007, July 19, pp. 67–72 (2007)
11. Hung-Yu, C.: SASI: A New Ultra-lightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. The IEEE Transactions on Dependable and Secure Computing 4(4), 337–340 (2007)
12. Rieback, M.R., Crispo, B., Tanenbaum, A.S.: RFID guardian: A battery-powered mobile device for RFID privacy management. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 184–194. Springer, Heidelberg (2005)
13. Juels, A., Rivest, R.L., Szydlo, M.: The blocker tag: selective blocking of RFID tags for consumer privacy. In: The 10th ACM conference on Computer and Communications Security, Washington D.C., USA, pp. 103–111 (2003)

Safe, Fault Tolerant and Capture-Resilient Environmental Parameters Survey Using WSNs

Gianni Fenu and Gary Steri

University of Cagliari, Computer Science Department,
Via Ospedale, 72, 09124 Cagliari, Italy
fenu@unica.it, steri@sc.unica.it

Abstract. Sensor networks are one of the first examples of pervasive computing, which is characterized by the massive use of increasingly smaller and powerful devices. A cloud of sensors arranged in a given environment is in itself a great source of data; accessing this source in order to extract useful information is not a trivial problem. It requires correct sensor deployment within the environment and a protocol for data exchange. We also have to bear in mind the problem of data and sensors security: sensors are often installed in areas difficult to protect and monitor. In this paper we describe SensorTree, a functioning model and a simulator for a network of wireless sensors installed on the sea surface to measure parameters useful for determining the weather situation.

Keywords: Key management in wireless/mobile environments, Data retrieval, Data security, Hashchain, Network resilience.

1 Introduction

Recent progress in micro-electromechanical systems (MEMS), wireless communications and digital electronics has allowed the development of increasingly efficient and cheap multifunctional sensor nodes capable of communicating with other sensor nodes through low-range wireless technologies.

A Wireless Sensor Network (WSN) is a set of sensor nodes arranged within or very close to the phenomenon to be observed [1]. It can comprise different kinds of sensors (e.g. seismic, magnetic, infrared, acoustic, radar) useful for monitoring many environmental conditions [2].

Sensors are able to "collaborate" and preliminarily process data, sending not just coarse data to collector nodes [1]. Thanks to this characteristic, sensor networks are suitable for many kinds of applications, such as medical, military and security.

The creation of these networks can require some techniques originally developed for wireless ad hoc networks. However, the protocols and algorithms directly derived from ad hoc networks do not fulfill sensor network purposes. We can argue this point outlining the main differences between these two kinds of networks [3]:

- the number of nodes in a sensor network can be much greater than in an ad hoc network;
- sensor nodes are deployed with high density;

- sensor nodes are prone to frequent failures;
- sensor network topology can change frequently;
- sensor nodes mainly use broadcast communications, whereas ad hoc networks are based on point-to-point communications;
- sensor nodes have very restricted power, computing and storage capabilities.

The last point is one of the main constraints of sensor networks; sensor nodes are equipped with limited power supply units and, generally, it is not possible to recharge or change them. This paper is concerned with the design and implementation of a model for a real sensor network application, that allows for all the conditions described above and provides secure communication scheme and data transfer, by means of multiple hashchains and techniques for node protection against external attacks.

2 Model Architecture and Network Functioning

SensorTree model has been conceived to obtain weather conditions by monitoring temperature, wind speed and luminosity at the sea surface close to a beach or a coastal area. It uses a multi-hop communication scheme with sensors identified by a unique ID and one Sink node always active. Each node should be able to measure the three weather parameters and send the data to Sink, which should be able to determine the situation throughout the entire monitoring area.

Each node lies within a rigorous level architecture defined over the stretch of sea to be monitored. The upper bound of the area (landwards) is delimited by Sink node and the lower bound (seawards) is defined by the number (indefinite) of levels. So the total number of levels is given by the number of nodes and their transmission range, which determines the level's height and width.

The operating area of a SensorTree network can be represented as a series of overlapping rectangles, as shown in figure 1. Level dimension has been chosen so that

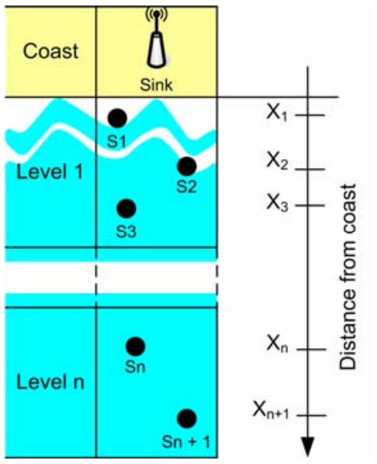


Fig. 1. Level architecture of SensorTree

transmission range of devices allows communication only between nodes at contiguous levels, in other words it is not possible to skip a level. The height of a level is a little more than half-transmission range and is equal to 5/4 of the width. Increasing level width risks compromising communication between a node of level n and a node of level n±1.

By virtue of this structure, each node resides only in one level (there are no overlapping areas) and has well defined coordinates within the area. We assume the devices are attached to a buoy and anchored to the sea bed, hence with negligible variations of their position.

2.1 Tree Construction

SensorTree network functioning is based on a tree structure which is built up along the levels of the monitoring area. The start-up of the entire survey system envisages a downward network growth starting from the Sink node; once Sink has been started, network construction begins turning on a sensor node (e.g. s1) in level 1. S1 notifies its presence sending a PowerOn request containing its position, battery level and other parameters[1].

Sink receives the request and sends to s1 a PowerOn reply communicating its identity, adding the node to its children list and marking it as "preferred" child. S1 immediately takes Sink as its root node (the only one for the first level).

Switching on another node (s2) in the same level, will produce nearly the same operation as above, but in addition:

- s1 does not reply to the PowerOn request sent by s2: requests sent by a node of the level n are handled only by nodes of levels n-1 and n+1;
- as soon as Sink receives s2 request, Sink adds s2 to its children list and checks its battery level: if it is higher than s1, then s2 will become the new preferred child.

The rules outlined above hold for any other node turned on in level 1; updating of the Sink's children list is very important.

Tree construction proceeds in the same way for the other levels, but from the second level onwards each node can choose between different roots. The choice of "preferred" root, as happens for children, is based on battery level, and variations therein can cause root changes.

2.2 Tree Maintenance and Convergence

The tree structure changes continuously; node battery level varies constantly, a few nodes can switch off or be destroyed and new nodes can join the network. So, a tree maintenance and convergence policy is required, that always guarantees the delivery of active nodes data to Sink.

Every T seconds (in simulations we chose T=10, but in similar contexts a longer time period can be used), each node sends to its neighbor nodes (children and roots, never nodes on its same level) a StatusRefresh notification containing battery level in addition to surveyed parameters. It is important to specify that T period starts a few

[1] In the simulation software, besides an address and the survey data, we used ports useful to different services and to test many instances of the application in the same machine.

instants after the node is switched on, so notifications are not synchronized. Obviously simultaneous notifications can occur but co-channel interference problems only arise for contiguous levels. The destination node (can be Sink too) receives the essential information indicating that its neighbor is still active and updates its lists using just received data, perhaps modifying preferred root and child.

Eventually some node fails; this means that it will no longer send StatusRefresh notifications. If a node stops sending send StatusRefresh notifications it will be deleted from its neighbor membership lists; if the node was a preferred node then preferred root and child must also be updated according to battery level of still active nodes. This mechanism has been implemented using a timer started at the same time as the last notification acceptance.

Should a node switch off or turn on before the timer has expired, probably no one would realize, except for the fact that when turned on again it sends a PowerOn request to detect neighbor nodes. In any case this request would be discarded (could be a duplicated ID) and the node would resume working as though nothing had happened.

2.3 Routing Tree

The tree construction described in the previous sections places much emphasis on node battery level and thus on estimating node life time. The choice of preferred roots and children is fundamental in determining routing paths used for data flow toward Sink.

From the above described rules it should be obvious that no node has a full view of the tree. Each node knows nodes on contiguous levels (its children and roots) but does not know what lies beyond. Only Sink can, at any time, perform a query on the tree and know all its details. Underlying nodes (not every one) can use the query to know next levels but they do not know anything about the previous ones.

In section 2.2 we stated that StatusRefresh notifications are not synchronized. This means that not every node has the same view of the tree at every instant. This problem is solved when it is required to know precisely the network status, that is when Sink requires a tree snapshot. This event triggers a data update of all sensors reachable from Sink, giving a full view of the monitoring area.

Tree tracking, like most of the operations performed on the tree, is recursive and is based on two simple steps:

1. Sink makes a StatusRefresh call to all its children which reply with a notification containing all data; Sink updates its children list and, if it is not empty, adds it to the procedure result;
2. The procedure is propagated recursively until the lower level (or until a tree interruption occurs) and the results returned to the upper level at each step.

This approach is not purely recursive because each root receives the first results simultaneously with propagation to next levels, not after reaching the end of the tree.

2.4 Nodes Querying

SensorTree simulator, in addition to tracking functionality, provides 4 modes to query the nodes. Queries do not require an immediate update of the lists: each node replies with data obtained during last refresh, such that no inconsistent data are sent. In fact, in these cases only one node is responsible for data sent to upper levels. Furthermore, the data provided are at the most T seconds "old" and this can be neglected for T values within reasonable ranges. Anyway instantaneous data can be obtained performing a refresh, as previously described, before querying the tree.

All query modes are based on checking children lists owned by each root and on query propagation to preferred child. Also in these cases the recursive technique is adopted with the same specification described in the previous section.

The first query mode involves all nodes and is useful for obtaining data for all sensors contained in the network. The results can be used, for example, for calculating average values of parameters for the whole tree. The preferred root of each level is responsible for the data transmitted to Sink; this query is able to detect possible duplicated nodes on the network.

The query on one node, detected using its ID, returns all data surveyed from that sensor: temperature (°C), wind speed (m/s), luminosity (lx). Moreover, it returns the level where the node is situated, its coordinates and its battery level. The presence of a duplicated ID is not detected because the procedure stops at the first matching node; however, retrieved information can be used to perform a consistency check with initial node distribution.

Another mode is the level query, which returns all data of the nodes in the level specified as query parameter. It can be used to verify if a level is reachable and detect tree interruptions.

Finally, the query on a point allows to obtain survey data in a particular point of the monitoring area, specified by its coordinates. Obviously not necessarily does a node exist at the exact point specified, therefore the data obtained pertain to the sensor closest to that point. Unlike the other queries, propagation does not follow the preferred child but the child closest to the requested point; along with parameter values survey distance is also determined.

2.5 Automatic Rejoining

It clearly emerges from the above sections that tree construction cannot skip any levels: turning on a node in a level not contiguous to the lower one, will not change the tree structure because this node is too distant to be reached. However it is possible to continue tree construction because turning on other nodes in the underlying levels follows the same rules explained so far, so the first non reachable node is the root of an independent sub-tree. This scenario can be also obtained when every node of a level stop working: the preferred child of the previous level and the preferred root of the next one have no links in that level and cannot communicate.

This kind of situation is inconsistent because sub-tree data cannot reach Sink, but is not so compromising: turning on a node in the empty level suffices to join the two trees. The mechanism used is very simple and allows the functioning of one or more independent trees; while the level is empty, data arriving to the root of the lower sub-tree are stored in a buffer so, if the period of time is enough short, survey data won't be lost. When a node in the empty level is turned on, it sends a PowerOn request that is satisfied by the preferred nodes of the contiguous levels and the sub-tree is automatically rejoined. At this point, the level previously empty, works like other levels; the steps are shown in figure 2 (the bold line between two nodes shows preferred root and child):

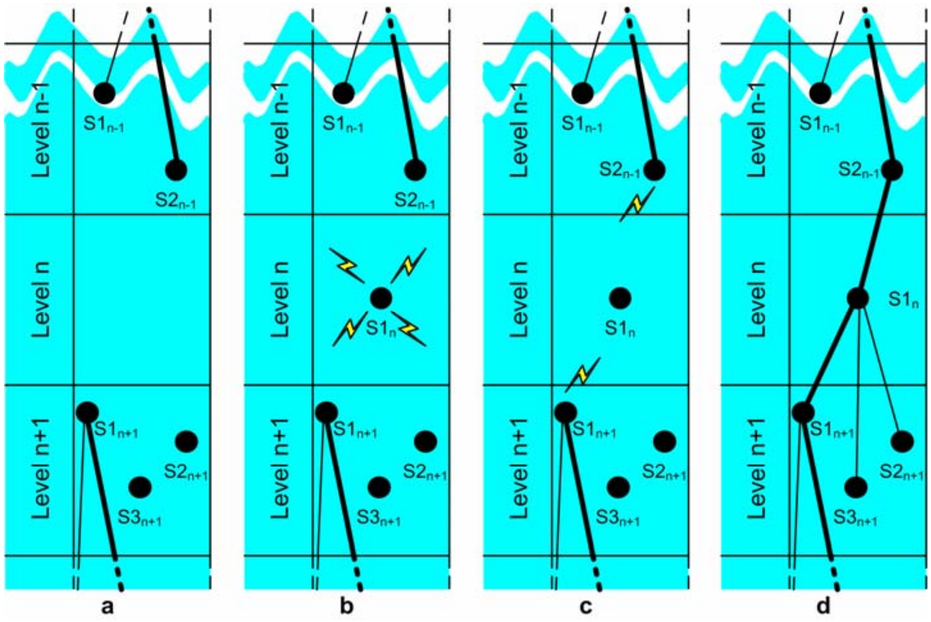


Fig. 2. Automatic rejoining steps. a) the level n is empty and there is an independent sub-tree starting from level n+1: $S1_{n+1}$ bufferizes survey data; b) $S1_n$ is turned on in the level n and sends a PowerOn request; c) $S2_{n-1}$ and $S1_{n+1}$ (respectively the preferred child and the preferred root of their levels) send a PowerOn reply; d) $S1_n$ is the preferred child of $S2_{n-1}$ and the preferred root of $S1_{n+1}$: the sub-tree is rejoined and data collected by $S1_{n+1}$ can reach Sink.

3 Data Integrity and Authenticity

Data flowing through the tree are not in themselves sensitive data and do not require specific policies to protect users' or third-party's privacy. However authenticity and integrity of data collected by Sink need to be guaranteed. For this reason, each message bound for Sink is accompanied by an authentication code computed by node's unique key, according to HMAC (keyed-hash message authentication code) standard:

$$H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text)) \tag{1}$$

In simulations we used SHA-1 algorithm (20 byte keys) and SHA-512 (64 bytes).

3.1 Nodes Initialization

Nodes need a key to authenticate communications directly addressed to Sink and another one to initialize a hashchain with its preferred root in order to send measured parameters. Since we do not use a public key infrastructure (PKI), prior to initializing communications with Sink we need to authenticate messages.

The initialization of a node is done during its installation, when it can load in main memory a key from a trusted source and use it to communicate with Sink (which obviously knows keys associated to each node) without saving it in a secondary memory. But in this way, only the first initialization of the node can be done and only being able to physically reach the node. Subsequent activations may be necessary in the event of malfunction, temporary exclusion from the tree or after attack by extraneous nodes.

In order to obtain this functionality each node has to be able to execute a hash function (e.g. SHA-1) and a secure deletion algorithm (e.g. Gutmann with appropriate passes only, Schneier, DoD-3 or other simpler ones). Moreover, the node has to store a list of keys to be used for subsequent activations (number of possible activations depends on number of keys but, then again, the device's life is limited by its battery duration). Memories considered here take into account dimensional specifications of the sensor network.

After the first initialization, a node that has to be remotely re-activated expects to receive from Sink the hash for the first key on the list (only Sink can compute it). As soon as this is received, the node computes the hash for the key and matches it with the one received, verifying that activation was actually started by Sink. At this point, the node loads the second key of the list in the main memory deleting it in secure mode from flash memory. The first key (which remained in main memory since first initialization) is now active and can be used to authenticate the messages directed to Sink. The activation sequence is shown in figure 3.

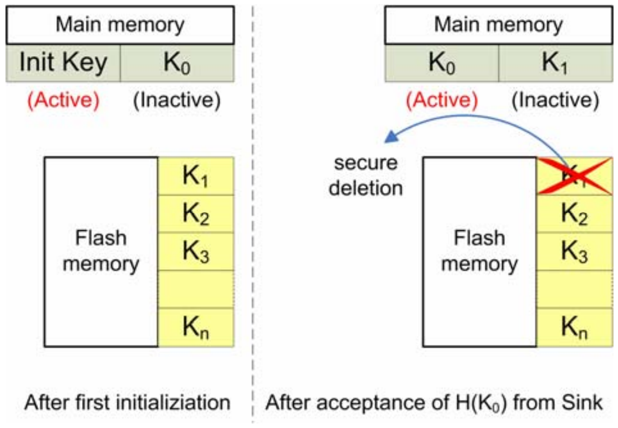


Fig. 3. Keys activation sequence

3.2 Data Collection and Hashchains

Data collection passes at each level through preferred root of the nodes, which is responsible for data sent to Sink. Obviously, in the first level, each node can send the data directly to Sink, which can therefore be authenticated using HMAC standard. Otherwise, each node can start a hashchain with Sink to guarantee data authenticity and integrity. This method is used for next levels of the tree, where each preferred root plays the role of Sink, in order to avoid individual node data being sent directly to Sink for authentication and the resulting generation of excessive traffic volumes.

In order to initialize the chain, each node, possible root, stores the hashes of a set of initialization keys owned by underlying nodes. The pool of keys is loaded in main memory as explained in the previous section for activation keys, while the hashes of the keys can be stored in the flash memory.

The chain is started by sending the key's hash to the preferred root together with first data hash; the next block contains first data, the hash of the second and so on. The chain can be interrupted and subsequent reinitializations cannot be done using the same hash. To solve this problem each root can store the hashes of different keys for each node, otherwise can expect to receive the nth hash of the key at first initialization, the n-1th at the second and so on, creating a one-time password authentication scheme. Root node storage and hashchain are shown in figure 4:

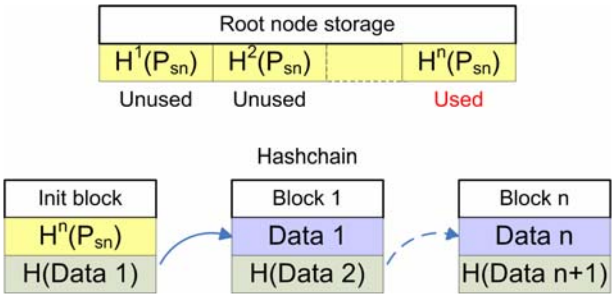


Fig. 4. Hashchains and one-time passwords

Using this mechanism a potential attacker wanting to enter the chain should be able to retrieve measurement data of next block using the hash of current block. Observing enough transmissions, an attacker will notice that some measurement data recur in the chain and analyzing the hashes can try to guess the content of the next block. To avoid this kind of attack, means ensuring that identical survey data generate different hashes, introducing additional data in the computation process. Introducing values obtained by a pseudo-random number generator (PRNG) can be useful but this does not significantly enhance security; inserting values obtained by a true-random number generator (TRNG) is the best solution. In simulations, the true-random values has been generated using parameter measurement intervals: the expiry time between one measurement and another is not extremely rigid and is influenced by hardware delays due to measurement devices and clock. The delays and the total time from last measurement are measured with an accuracy higher than interruptions (e.g. milliseconds or nanoseconds) and added to data packet (which also contains unique IDs for each node). The probability that an attacker guesses the exact instant of data measurement and sending is really low.

Once the preferred root has collected data from each child and verified their authenticity, it sends them directly to Sink using a JumboData packet that is forwarded on the routing tree path. This packet does not contain authentication data of individual nodes but only an authentication code computed by the root according to HMAC standard.

4 Capture-Resilience

The most difficult problem to solve in this kind of application is the physical protection of the nodes. An attacker can steal them, copy their content, clone them or exchange them with other devices. In order to protect the nodes we need to ensure that even if the attacker does steal the node, he is unable to extract useful information for accessing the network.

In the nodes activation procedure and for using the keys for measurement data sending, we have chosen to store the active key and the next key of the activation pool in the main memory (RAM) alone, performing a secure deletion from the secondary memory (flash memory) of the loaded keys. The latter step is very important because, even after a flash memory erase operation, the transistors do not return fully to their initial state, thereby allowing the attacker to distinguish between previously programmed and not programmed transistors, and thus restore information from erased memory [4].

We also have to prevent the attacker from accessing the main memory content, for example by creating tamper-protection for the node which forces node shutdown in the event of intrusion. But shutdown in itself is not sufficient, because RAM memories are also prone to data retention problems: using a key for a long period of time can cause long-term retention effects in the memory [5], leaving useful traces for restoring information. Therefore memory cells should not store data for too long a period of time (e.g. performing a bit flipping) and in the event of tamper detection it is necessary to apply a reverse current which reverses the electromigration stress due to long-term retention [5]. In this way node theft and access to its content render the node unusable for joining the network.

In our tests and simulations we verified the functioning of solutions proposed in previous sections proving that latency times of joining procedures and query propagations allow a correct data delivery to Sink node; for this purpose we also considered latencies introduced by capture-resilience measures for keys activation sequences and nodes initialization.

5 Conclusion

The model described above has allowed to develop a simulator which fulfils all requirements for the correct functioning of SensorTree network. The proposed model is also the cornerstone for an ongoing development project for testing and creating the network.

The use of the tree structure enables efficient communication between the nodes minimizing the paths to Sink. In spite of this advantage, the tree structure introduces geometrical constraints on level dimensions;: However, this problem can be solved by introducing intra-level communications and designating some groups of nodes as landmarks. Intra-level communication can also be useful for differentiating data collection modes and for simplifying pure recursive operations on the tree.

Level subdivision enables critical network areas to be identified and to act simply on those areas, maintaining a regular functioning of the other levels (even with independent trees). Even simply adding one node to the network can create a critical situation when the number of nodes is very large. This is the reason why it may be necessary to develop a control protocol to handle this situation, to quickly solve tree interruptions and to allow Sink node mobility.

The choice to rely on hash algorithms and shared keys for message authentication and integrity well matches sensor node capabilities, whose most recent commercial versions have barely sufficient resources to operate on asymmetric key encryption algorithms [6]. This choice has also made it possible to reduce data overheads introduced by encryption algorithms, especially for very large data blocks. However, the use of cryptography (even if symmetric) could be useful for ensuring greater protection of some communications, for simplifying key redistribution and for ensuring data privacy when necessary.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38, 393–442 (2002)
2. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: scalable coordination in sensor networks. In: ACM MobiCom 1999, Washington, pp. 263–270 (1999)
3. Perkins, C.: Ad Hoc Networks. Addison-Wesley, Reading (2000)
4. Skorobogatov, S.: Data Remanence in Flash Memory Devices. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 339–353. Springer, Heidelberg (2005)
5. Gutmann, P.: Data Remanence in Semiconductor Devices. In: 10th USENIX Security Symposium Proceedings, Washington (2001)
6. Tan, H., Jha, S., Hostry, D., Zick, J., Sivaraman, V.: Secure Multi-hop Network Programming With Multiple One-way Key Chains. In: WiSec 2008, Alexandria (2008)
7. Çamtepe, S.A., Yener, B.: Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks. IEEE/ACM Transactions on Networking 15(2) (2007)
8. Akyildiz, I.F., Vuran, M.C., Akan, O.B.: A Cross-Layer Protocol for Wireless Sensor Networks. In: Conference on Information Science and Systems (CISS 2006), Princeton (2006)
9. Gutmann, P.: Secure Deletion of Data from Magnetic and Solid-State Memory. In: 6th USENIX Security Symposium Proceedings, San José (1996)
10. Giannotti, F., Pedreschi, D.: Mobility, Data Mining and Privacy Geographic Knowledge Discovery. Springer, Heidelberg (2008)
11. Tracy, L.T., Roy, S.: A reservation MAC protocol for ad-hoc underwater acoustic sensor networks. In: WUWNet 2008, San Francisco (2008)
12. Badia, L., Mastrogiovanni, M., Petrioli, C., Stefanakos, S., Zorzi, M.: An optimization framework for joint sensor deployment, link scheduling and routing in underwater sensor networks. In: WUWNet 2006, Los Angeles (2006)

SAVAH: Source Address Validation with Host Identity Protocol

Dmitriy Kuptsov and Andrei Gurtov

Helsinki Institute for Information Technology
Helsinki University of Technology
{dmitriy.kuptsov,gurtov}@hiit.fi

Abstract. Explosive growth of the Internet and lack of mechanisms that validate the authenticity of a packet source produced serious security and accounting issues. In this paper, we propose validating source addresses in LAN using Host Identity Protocol (HIP) deployed in a first-hop router. Compared to alternative solutions such as CGA, our approach is suitable both for IPv4 and IPv6. We have implemented SAVAH in Wi-Fi access points and evaluated its overhead for clients and the first-hop router.

Keywords: Security, Authentication, LAN, HIP.

1 Introduction

Routing of packets in the Internet is based on the destination IP address. It is hard to identify the source of the packet. Attackers can easily spoof the source IP address of a packet; hosts under Denial-of-Service (DoS) attacks cannot trace the originators. Another negative result can be a blocked service for a given source address, which can be the address of a good host compromised by an attacker.

These issues forced the source address validation to become an urgent problem in networking research. Several existing solutions are based on cryptographic authentication, traceback, and filtering. In this paper, we propose a solution called *Source Address Validation Architecture with Host Identity Protocol (HIP)* (SAVAH), which involves cryptographic authentication and filtering based on the host identifiers. This method integrates with a source address validation architecture (SAVA) [1], and acts as an alternative to proposed method of validating source addresses on the edge (or first-hop) router using Cryptographically Generated Addresses (CGA).

Host Identity Protocol (HIP) is a new security protocol which integrates host locator/identifier split, mobility, and multihoming. It was specified by IETF [10,12,11,8,9,14,13,16]. Possessing important properties such as a secure and efficient session key negotiation and self generated host identifiers for authentication, HIP can help to solve the problem of source address validation.

We believe that SAVAH can be deployed in a large-scale network and integrated with existing solutions deployed on different level of granularity. In general, we think that HIP-based source address validation can become a replacement for a CGA approach [1]. We also think that it provides more security than any of the existing solutions for the source address validation in a local network, as it relies on a cryptographically secure protocol.

The paper is organized as follows. Section 2 gives an overview of Host Identity Protocol. Section 3 summarizes related work. In Section 4, design and implementation of SAVAH is described. In Section 5, we discuss integration of our proposal with general SAVA architecture. Performance measurements are given in Section 6 and Section 7 concludes the paper.

2 Background on Host Identity Protocol

The existing Internet architecture was designed for stationary hosts and faces many non-trivial challenges today with the growing number of mobile terminals. Currently, there are two namespaces used globally by the Internet services and applications, domain names and IP addresses. IP addresses serve the dual role in the Internet being both end host identifiers and topological locators. This general principle does not allow hosts to change their location without breaking ongoing transport protocol connections that are strictly bound to IP addresses.

The Host Identity Protocol (HIP) [8,6] was proposed to overcome the problem of using IP addresses for host identification and routing. The idea behind HIP is based on decoupling the network layer from the higher layers in the protocol stack architecture (see Figure 1).

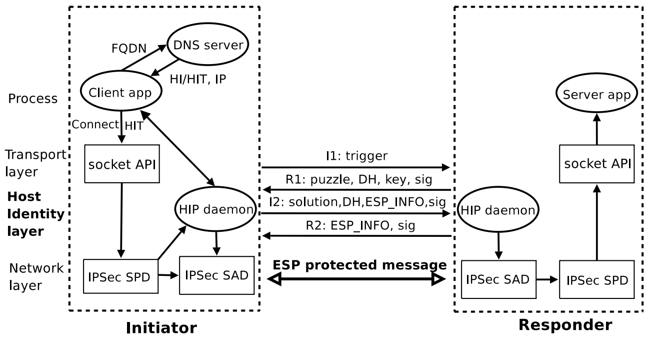


Fig. 1. HIP architecture

HIP defines a new global name space, the Host Identity name space, thereby splitting the double meaning of IP addresses. When HIP is used, upper layers do not anymore rely on IP addresses as host names. Instead, Host Identities are used by the transport protocol for establishing connections. IP addresses at the same time act purely as locators for routing packets towards the destination. For compatibility with IPv6 legacy applications, Host Identity is represented by a 128-bit long hash, the Host Identity Tag (HIT).

HIP offers several benefits including end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, mobility and multihoming support.

3 Related Work

There are three main methods for source address validation: cryptographic authentication, ingress/egress packet filtering, and various traceback techniques.

Cryptographic authentication appears a promising solution which brings strong security properties for authenticating originator of the network communication. IPSec [7] is one example, which allows secure end-to-end communication. Wu et al. [15] point out that IPSec depends on global deployment of PKI infrastructure. HIP [6,8] in combination with HIP-enabled firewall, in turn, can provide basic functionality for source address authentication and validation. One possibility will be covered in Section 4.

Another approach is SPM [5] designed for authorization of neighboring autonomous systems (AS). This approach provides solution for source address authorization and relies on cryptographic properties.

Filtering of malicious packets containing wrong source addresses can be considered a simple solution. Unlike solutions based on cryptographic properties, it is straightforward to deploy, but less secure approach. Examples can be ingress filtering, SAVE protocol [4] and HCF filtering [3].

4 SAVAH Design and Implementation

We considered three possible approaches when designing the source address validation mechanism using HIP.

1. First, a so-called pure HIP communication with HIP firewall in-between. This requires that all communicating peers support HIP. Then the HIP firewall (which should be deployed on an edge of a local network) tracks base exchange signaling packets from all hosts that try to communicate with the hosts outside the local network and only lets through the packets with valid HITs and IP addresses. Later, data is sent in ESP encapsulated packets so that the firewall can check the SPI values of the packets and drop those that do not match a previously seen base exchange. This approach requires a large-scale deployment of HIP protocol.
2. Secondly, we assume that hosts inside the local network support HIP with our extension. Then the HIP base exchange can replace a CGA approach [15]. This case is the main focus of the article and will be discussed in detail.
3. Finally, HIP tunneling approach can be used. It is less efficient in performance, but can provide better security and mobility support to wireless clients. The client creates a tunnel between itself and the SAVAH router using HIP base exchange and SAVAH extension. Later all traffic is forwarded through this tunnel to the Internet. There were previous studies on tunneling the traffic to home router in PISA [2].

4.1 SAVAH Architecture

SAVAH targets to solve the source address validation problem in the edge router of the local network serving as a default gateway. SAVAH architecture is composed of two main components:

– A SAVAH-enabled client, which is a combination of a HIP daemon and a firewall in a client mode supporting SAVAH extension.
– A SAVAH-enabled router running the HIP daemon and the firewall but in a server mode.

DHCP server can be considered the third component in our architecture. Its main role in the network is to offer particular configuration for SAVAH aware hosts. For instance, DHCP can provide the default gateway IP address as well as a HIT of the SAVAH router. Availability of a proper configured DHCP server can help to solve the problem of opportunistic mode as discussed later in this section. However, we consider DHCP as an optional component in the network.

SAVAH aware clients can be configured manually, meaning that the IP address and the HIT of the SAVAH router can be setup by a system administrator prior to any communication. However, manual configuration can be a tedious task in a large scale network. As the third option, the SAVAH-enabled client can discover the SAVAH router using the opportunistic mode.

The message sequence diagram for SAVAH registration and further authentication process is shown in Figure 2. It involves three entities in the local network to perform the source address validation: the SAVAH client, the SAVAH router, and the DHCP server (optionally). The receiver in Figure 2 is playing the role of a legacy peer, i.e., it may or may not support HIP. Of course, if both communicating peers support HIP, whole scenario requires only a normal HIP-enabled firewall to filter the traffic based on HITs.

Since the SAVAH router is the first-hop router, it should be placed on the edge of the local network and serve as a default gateway. If the default gateway and the SAVAH router are not placed physically on the same node it is meaningless, because all network traffic flowing through the SAVAH-unaware router would not contain the SAVAH option and would be discarded.

4.2 SAVAH Router Discovery

Unless the DHCP server offers a HIT of the SAVAH router during the address assignment phase, the SAVAH client is obliged to discover the presence of the SAVAH service automatically. Otherwise, the client is aware of the HIT and the IP address of the SAVAH router and can register to the service directly. It is also possible to preconfigure the client and specify the IP address and HIT of the default gateway.

Optionally, the presence of a SAVAH-unaware DHCP means that the client will obtain only the IP address of the default gateway. No prior knowledge of the SAVAH router's HIT forces the client to discover the service using the HIP

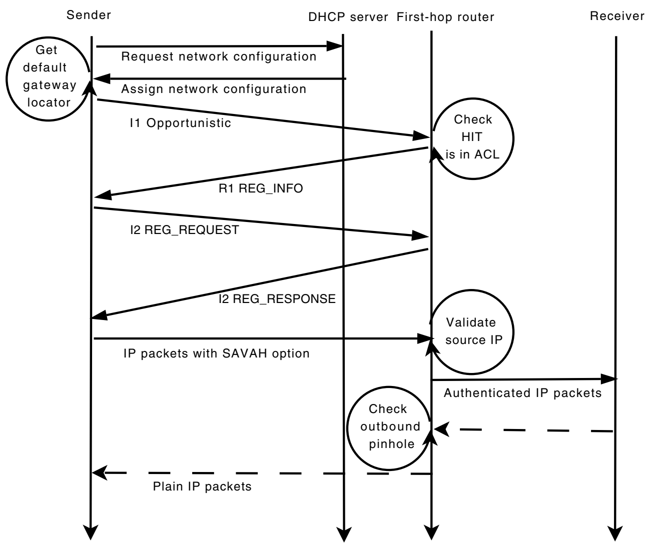


Fig. 2. SAVAH service registration and authentication process

opportunistic mode [6] and a set of usual procedures for HIP service registration [11].

The drawback of such broadcasting is that in the opportunistic mode the client is unaware of the HIT of SAVAH router and any node can thus pretend to be a valid router. This resembles a similar problem with SSH when connecting to unknown hosts. Hence, the opportunistic mode should be used only in a trusted environment.

To trigger a registration in opportunistic mode, the SAVAH client requests from the system the default gateway IP address and sends an I1 packet with the destination HIT as a hashed source IP address. On the other side, the default gateway running SAVAH in a server mode responds with an R1 packet containing an offer for available services in REG_INFO parameter.

Upon receiving the R1 packet the client chooses the supported services and responds with an I2 packet containing a REG_REQUEST parameter to SAVAH server. If REG_INFO parameter does not contain the SAVAH service offer, the client completes base exchange normally and afterward falls back to normal communication, i.e., the SAVAH mode is not supported in this network.

Finally, depending on the setup of the SAVAH router, it either grants or denies the service to a client in an R2 packet with a REG_RESPONSE or REG_FAILED parameter. Receiving a REG_FAILED parameter in an R2 message during the base exchange or experiencing a timeout in an I1 state means that either the default gateway does not support SAVAH extension or that HIP daemon with SAVAH extension is not running at all.

This situation should indicate the SAVAH client to fallback to normal communication, i.e., the packets to be forwarded through the default gateway would not contain any authentic information. As an opposite result, if the SAVAH router grants the service to the registering client, both parties will posses a shared secret key.

4.3 Packet Authentication

For performance reasons, the keys in use for authentication (i.e., HMAC keys) in IPSec were selected to authenticate the source addresses. Depending on a setup, symmetric cryptography can be selected as well.

Filtering on HITs can be applied to ensure that the peer trying to register to the SAVAH service is legitimate. This filtering can enforce to either grant or deny the SAVAH service to the registrars. Firewall rules in the HIP-enabled firewall control such decisions. By default, all packets with an unknown identifier are dropped.

After completion of the HIP base exchange, the SAVAH router adds the source IP address of the host to a database. This works if no record with such IP address were already present in the database. If a record with such IP address already exists, it is likely that the host is trying to spoof someone else address and the packet should be dropped nor any state added. Moreover, this incident can be logged and reported for further analysis.

If the host experiences an address change, then the record is replaced with a new IP address. To ensure the validity of the host, a HIP UPDATE packet has to be received and handled properly. This will guarantee that the host indeed the one it is claiming to be. The record should be removed from the database upon a timeout or when the host removes a security association (e.g., a HIP CLOSE packet is received from the corresponding host). To ensure that the client is alive, the SAVAH router can also send heartbeats to the client without waiting for the timeout.

After the secret keys are established by means of the HIP base exchange, the SAVAH client may communicate with the nodes outside of the local network by including an authenticated source IP address in each packet. Current implementation has two options to deliver the authenticated hash value to the router. The first approach is to replace the original source IP address of each packet with truncated result obtained from $HMAC(key|\{P\})$ operation, where $\{P\}$ is the packet to be transmitted. We have chosen the packet value as a feed to the HMAC function to introduce a simple protection mechanism from replay attacks.

This approach has some drawbacks. First of all, for IPv4 networks the size of authentication value will be only 32 bits. Secondly, that would decrease the performance since additional address translation would be required on the router. Finally, that method will require additional changes to the router to recognize locally routed traffic.

A different way to carry the authenticated source IP address to SAVAH router is to store it in an IP option. Since the SAVAH router is the first-hop router and all SAVAH related options are striped out on the forward direction, this ensures that no modifications are required for other routers on the path. Placing the authentication value in the IP option can have certain advantages. First of all, this approach slightly optimizes the performance of the architecture. Secondly, stronger security can be achieved by increasing the length of the authentication value. We suggest to keep this length within wise bounds, since it affects the size of the actual payload.

4.4 Source Address Validation

To authenticate the packet source address, the following algorithm is used. On the router side, each packet in forward direction is checked for the SAVAH IP option. If found, the router compares the value of the option against truncated 128-bit result from $HMAC(key|\{P\})$ operation, where $\{P\}$ is the packet to be forwarded. If matches, the SAVAH option is striped out and the packet is re-injected to the network. If not, the pinhole is searched for a given source and destination IP addresses. If the pinhole is found, the packet is said to be an inbound packet for previously authenticated outbound communication. Finally, if both fail the packet is dropped.

If the tunneling approach is used, then the authentication succeeds if the SAVAH router can successfully decrypt the ESP packet and resend encapsulated in ESP original packet to its final destination. Unlike in the lightweight approach for inbound traffic, the SAVAH router in a tunnel mode should properly encrypt and tunnel the packet to the corresponding mobile node.

Storing a mapping between the identity and the source IP address of each accessing client on the SAVAH router, allows to keep track for the duplicate and spoofed IP addresses, and account each packet traversing the router.

Whether the network access is controlled by the ACL or not, spoofing of the source IP addresses is eliminated as there can be only one mapping between one particular HIT and the IP address. The mapping can be updated or removed once the corresponding UPDATE or CLOSE packet arrives and validated by the SAVAH router. Even when no ACL is maintained, the client cannot forge the source IP address because for each newly self-assigned IP address and/or generated HIT the user needs to complete a four-way HIP handshake with the SAVAH router.

If the address is spoofed with an address of the same subnetwork, the SAVAH router will detect a duplicate address use (because of a locally stored mapping table), and will prohibit the registration to the service. As an additional security action, such activity can be logged and reported. Moreover, non-repudiation property of HIP allows to take a counter measure against such users. If the client tries to spoof the IP address with an address of a network other than the local, such packets should be filtered out with ingress filtering deployed on the SAVAH router. Hence, spoofed packets will be dropped and the user will be logged as malicious and banned from using the network.

For the reasons discussed in the previous paragraph, several address spoofing and denial-of-service attacks are eliminated. For instance, amplifying attacks including various redirect attacks (e.g., DNS redirect, smurf, fraggle, etc.), DoS attacks (those that rely on spoofing the source address), and SYN flood attacks (since most of the SYN flood attacks require to generate TCP SYN packets with a spoofed source address) become hard to launch.

5 Deployment and Integration with General SAVA Architecture

We are planning to pilot our architecture in a large-scale IPv6 network in Tsinghua University and CERNET2 in China. Our source address validation is placed on the border between local network and the first autonomous system. Instead of CGA mechanism for source address validation [1], HIP with SAVA extension is used as shown in Figure 3.

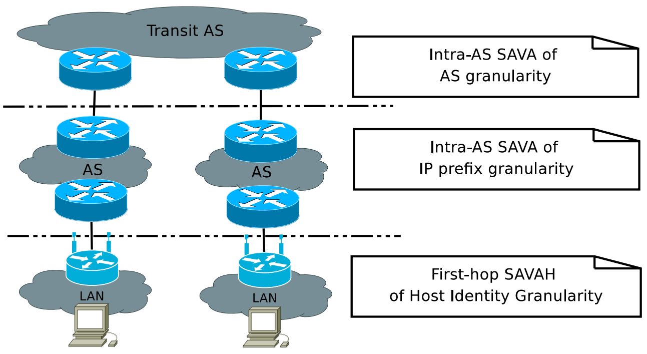

Transit AS
Intra-AS SAVA of AS granularity
AS
AS
Intra-AS SAVA of IP prefix granularity
LAN
LAN
First-hop SAVAH of Host Identity Granularity

Fig. 3. Integrating SAVAH with inter-AS source address validation

The key strength of our proposal is that in addition to basic source address validation our implementation adds support for mobility, multihoming, data integrity and encryption. Another advantage is that accounting can be maintained easily since the network access is controlled based on valid registered HITs. Although this is not necessary to validate the source address, it can be considered as a plus in large-scale networks. However, as with CGA mechanism, our architecture requires modifications to hosts residing in the local network to pass the authentication procedure.

Figure 4 shows how SAVAH can be deployed and used in a public wireless network, for instance in PanOULU [18] or Tsinghua University campus network, to validate, authenticate, and account network traffic. The figure describes possible step-by-step procedure for adding an unknown HIT to ACL:

1. First, a client tries to register with a SAVAH router using the procedure described in Section 4.
2. In case the HIP firewall does not find the client HIT in the ACL, it drops all packets except for HTTP traffic which is redirected to the SAVAH registration HTTP server. Otherwise, the client successfully completes the HIP base exchange and is permitted to use the network.

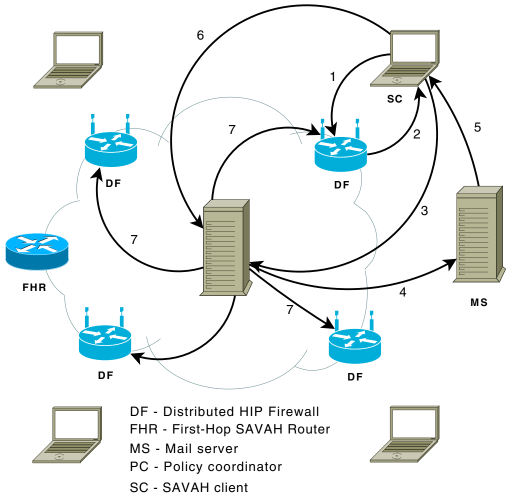


Fig. 4. A HIT registration procedure

3. A web registration form is sent to the client browser. The client provides his email and HIT and submits the form. As an alternative the client can identify itself by some other mechanism, for instance using Internet banking.
4. The SAVAH registration server checks if the submitted email belongs to the list of allowed domains (pre-configured database). It generates the authentication link and sends it to the given email address.
5. The client retrieves the email.
6. The client clicks the link in the email and thus authenticates the previously submitted HIT.
7. The authenticated HIT is distributed to all SAVAH routers and HIP firewalls.

6 Performance Evaluation

In the experiential setup, we used a wireless access point and a laptop. The wireless access point was running a modified version of OpenWRT Linux distribution [17], a HIP daemon, and a firewall in a SAVAH server mode. The laptop was running Ubuntu Linux and HIP in a SAVAH client mode.

We assume that our network is trusted, hence we use an opportunistic mode to discover the SAVAH router presence in the network. Hardware characteristics of the SAVAH router and client are shown in Table 1. Our testbed supports both IPv4 and IPv6 network stacks and we can evaluate SAVAH extension in both configurations.

We performed measurements of packet processing time in the router and the client. Collected results are shown in Figure 5 and Figure 6 for the router and client correspondingly. The expected time for packet processing in both ends

Table 1. Test hardware

Parameter	First-hop router	Laptop
CPU	533 MHz	Dual core 2.4 GHz
RAM	128 MB	3GB
Wireless	Atheros BG	Intel AGN

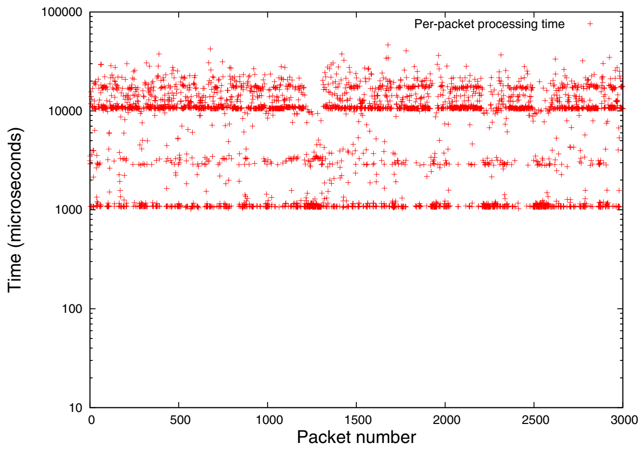


Fig. 5. SAVAH packet processing time in the first-hop router (Avila board)

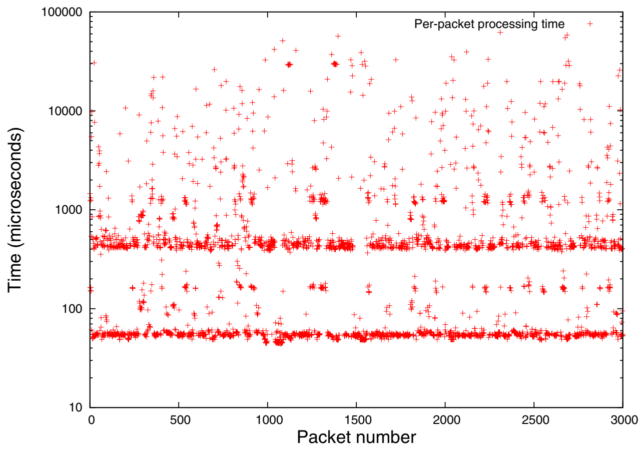


Fig. 6. SAVAH packet processing time in a client (laptop)

(i.e., for router and client) is $9ms$ and $1ms$. In theory, these results should allow to process on the average 110 packets per second in the router and around 1000 packets in the client. Hence, we estimate throughput of 1.3Mbps and 13Mbps in the router and in the client (assuming the Ethernet MTU 1500B).

Such performance is not sufficient for a router on the edge of a network with heavy traffic. Since our implementation is only a proof-of-concept, its optimization should help to overcome this hurdle. On the other hand, implementation can be faster in using RSA or DSA signatures [2]. For instance, the time required to check one DSA signature was about 63.7 *ms*. However, by compiling the prototype without debugging information and with several optimizations, the performance can be increased by $150 - 200\%$.

The memory costs are insignificant in the SAVAH mode. We have tested SAVAH extension during long period of time by streaming video files. We have noticed that memory usage on the SAVAH router stayed below 26MB boundary. On a device such as the Avila board (used as a router), this is not significant.

The CPU usage showed 100% load when heavy traffic is streamed through the router. The scheduler releases all CPU cycles to a demanding application. In our case, the HIP daemon and firewall were the only applications in an active state. Thus, this does not necessarily leads to a decrease in performance of other applications. Instead, 100% usage of CPU tells that the application is a resource hungry. However, memory copy operations required to strip out the SAVAH option are currently inefficient and can be optimized. On the other hand, calculating HMAC does stress the CPU and a packet is processed faster than if any other (symmetric or asymmetric) cryptography would have been used.

7 Conclusions

In this paper, we proposed a secure mechanism for validating source addresses and authenticating hosts in the local network. In SAVAH, local hosts use the extended Host Identity Protocol (HIP) to connect to legacy Internet hosts through a first-hop router which authenticates users.

SAVAH architecture involves three network entities, a network configuration server (DHCP), a client, and a router supporting extended HIP for host authenticating and source address validation. We implemented the system by re-flashing firmware of a wireless access point with OpenWRT Linux and HIP protocol supporting SAVAH extension.

Performance evaluation showed that SAVAH wireless access points can validate source addresses and provide reasonable protection against address spoofing. In addition, SAVAH offers stronger cryptographic properties including authentication, authorization, accountability, non-repudiation, and consistency. Our approach enables host mobility and multihoming. In contrast to CGA-based approach, SAVAH supports both IPv4 and IPv6, and is a patent-free technology.

References

1. Wu, J., Ren, G., Li, X.: Source Address Validation: Architecture and Protocol Design. In: IEEE International Conference on Network Protocols, pp. 276–283. IEEE Computer Society Press, Los Alamitos (2007)
2. Heer, T., Li, S., Wehrle, K.: PISA: P2P Wi-Fi Internet Sharing Architecture. In: 7th IEEE International Conference on Peer-to-Peer Computing, pp. 251–252. IEEE Computer Society Press, Los Alamitos (2007)

3. Jin, C., Wang, H., Shin, G.K.: Hop-count filtering: an effective defense against spoofed DDoS traffic. In: 10th ACM conference on Computer and communications security, pp. 30–41. ACM Press, New York (2003)
4. Li, J., Mirkovic, J., Wang, M., Reiher, P., Zhang, L.: SAVE: Source address validity enforcement protocol. In: 21st Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1557–1566. IEEE Press, Los Alamitos (2002)
5. Bremler-Barr, A., Levy, H.: Spoofing Prevention Method. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 536–547. IEEE Press, Los Alamitos (2005)
6. Gurtov, A.: Host Identity Protocol (HIP): Towards the Secure Mobile Internet. John Wiley and Sons Publishing, Chichester (2008)
7. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol, IETF, RFC 2401 (1998)
8. Moskowitz, R., Nikander, P.: Host Identity Protocol Architecture, IETF, RFC 4423 (2006)
9. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Experimental Host Identity Protocol (HIP), IETF, RFC 5201 (2008)
10. Jokela, P., Moskowitz, R., Nikander, P.: Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP), IETF, RFC 5202 (2008)
11. Laganier, J., Koponen, T., Eggert, L.: Host Identity Protocol (HIP) Registration Extension, IETF, RFC 5203 (2008)
12. Laganier, J., Eggert, L.: Host Identity Protocol (HIP) Rendezvous Extension, IETF, RFC 5204 (2008)
13. Nikander, P., Laganier, J.: Host Identity Protocol (HIP) Domain Name System (DNS) Extension, IETF, RFC 5205 (2008)
14. Nikander, P., Henderson, T., Vogt, C., Arkko, J.: End-Host Mobility and Multihoming with the Host Identity Protocol (HIP), IETF, RFC 5206 (2008)
15. Bi, J., Wu, J., Yao, G.: A CGA based Source Address Authorization and Authentication (CSA) Mechanism for First IPv6 Layer-3 Hop: draft-bi-savi-csa-00, IETF, Internet Draft (2007)
16. Nikander, P., Melen, J.: A Bound End-to-End Tunnel (BEET) mode for ESP: draft-nikander-esp-beet-mode-09, ITEF, Internet Draft (2008)
17. OpenWRT Web site, http://www.openwrt.org
18. PanOULU Public WLAN Network, http://www.panoulu.net

Secure Service Invocation in a Peer-to-Peer Environment Using JXTA-SOAP

Maria Chiara Laghi, Michele Amoretti, and Gianni Conte

Information Engineering Department, University of Parma, 43100 Parma, Italy
laghi@ce.unipr.it, {michele.amoretti,gianni.conte}@unipr.it
http://dsg.ce.unipr.it

Abstract. The effective convergence of service-oriented architectures (SOA) and peer-to-peer (P2P) is an urgent task, with many important applications ranging from e-business to ambient intelligence. A considerable standardization effort is being carried out from both SOA and P2P communities, but a complete platform for the development of secure, distributed applications is still missing. In this context, the result of our research and development activity is JXTA-SOAP, an official extension for JXTA enabling Web Service sharing in peer-to-peer networks. Recently we focused on security aspects, providing JXTA-SOAP with a general security management system, and specialized policies that target both J2SE and J2ME versions of the component. Among others, we implemented a policy based on Multimedia Internet KEYing (MIKEY), which can be used to create a key pair and all the required parameters for encryption and decryption of service messages in consumer and provider peers running on resource-constrained devices.

Keywords: service, peer-to-peer, security.

1 Introduction

Peer-to-peer (P2P) is clearly arising as a disruptive paradigm enabling highly scalable decentralized applications based on resource sharing. A peer-to-peer architectural model defines application-level protocols and policies for the construction and maintenance of an overlay network, where each peer node is itself a resource, being discoverable and contributing to the operation of the whole system.

While for many years the focus of P2P research has been on performance optimization of content sharing applications, there is now a growing interest for service-oriented P2P systems, in which peers offer consumable resources, *i.e.* resources that cannot be acquired (by replication) once discovered, but may only be directly used upon contracting with their hosts [11].

The convergence of service-oriented and peer-to-peer architectures requires standards for (1) overlay network construction and maintenance, (2) message routing among peers, (3) service advertising, discovery and interaction, (4) security. While SOAs have sound, widely accepted standard protocols covering points (3) and (4), for example within Web Service technologies, on the other hand P2P systems lack of unified views and solutions.

Among others, Sun MicroSystems' JXTA is probably the most advanced attempt to provide a standard set of P2P protocols [12], covering all listed points to a certain degree. JXTA protocols should allow a vast class of networked devices (smartphones, PDAs, PCs and servers) to communicate and collaborate seamlessly in a highly decentralized fashion. The JXTA framework defines a naming scheme, advertisements, peergroups, pipes, and a number of core policies, while the JXTA middleware implements the specifications in Java and C++.

In a previous work [13] we introduced the JXTA-SOAP component, which is an official extension for JXTA, enabling Web Service sharing in peer-to-peer networks. Here we provide more implementation details, focusing on security aspects, for both J2SE and J2ME versions of JXTA-SOAP.

The paper is organized as follows. In section 2 we present related work on peer-to-peer security, focusing on service interaction issues. In section 3 we describe the architecture of JXTA-SOAP, with reference to adopted technologies. Section 4 illustrates secure service invocation mechanisms, with different policies depending on the characteristics of the device that hosts the peer. Finally we conclude the paper with a summary and an outline of future work.

2 Background

Making peer-to-peer systems secure is a significant challenge. A malicious node might give erroneous responses to requests, both at the application level, returning false data, or at the network level, returning false routes and partitioning the network. Moreover, the P2P system must be robust against a conspiracy of a malicious collective, *i.e.* a group of nodes acting in concert to attack reliable ones. Attackers may have a number of goals, including traffic analysis against systems that try to provide anonymous communication, and denial of service against systems that try to provide high availability.

Security attacks in P2P systems are classified into two broad categories: passive and active [5]. Passive attacks are those in which the attacker just monitors activity and maintains an inert state. The most significant passive attacks are:

– *Eavesdropping*, which involves capturing and storing all traffic between some set of peers searching for some sensitive information (such as personal data or passwords).
– *Traffic analysis*, where the attacker not only captures data but tries to obtain more information by analyzing its behavior and looking for patterns, even when its content remains unknown.

In active attacks, communications are disrupted by the deletion, modification or insertion of data. The most common attacks of this kind are:

– *Spoofing*, in which one peer impersonates another, or some outside attacker transforms communications data in order to simulate such an outcome.

– *Man-in-the-middle*, where the attacker intercepts communications between two parties, relaying messages in such a manner that both of them still believe they are directly communicating. This category includes data alteration between endpoints.
– *Playback or replay*, in which some data exchange between two legitimate peers is intercepted by the attacker in order to reuse the exact data at a later time and make it look like a real exchange. Even if message content is encrypted, such attacks can succeed so long as duplicate communications are allowed and the attacker can deduce the effect of such a repeat.
– *Local data alteration*, which goes beyond the assumption that attacks may only come from the network and supposes that the attacker has local access to the peer, where he can try to modify the local data in order to subvert it in some malicious way.

To cope with these attacks, security policies adopted at the overlay P2P network level usually consist of key management, authentication, admission control, and authorization. These are the strategies we took into account for securing consumer-to-service communication in JXTA-SOAP.

Transport-level security is mostly based on Secure Sockets Layer (SSL) and on its successor Transport Layer Security (TLS) that runs beneath HTTP, the most popular protocol for Web Services. HTTP is an inherently insecure protocol because all information is sent in clear text between unauthenticated peers over an insecure network. Transport-level security can be applied to secure HTTP; the purpose of TLS is to create a secure pipe between two web servers. Authentication occurs at the time the secure pipe is created, and confidentiality and integrity mechanisms are applied only while the message is in the secure pipe. When running over an TLS-protected session, the server and client can authenticate one another and negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. At the receiving endpoint the messages are decrypted by the server.

Once a Web Service transmits a message it does not always know where that message will be traveling to, and how it will be processed before it reaches its intended destination [14]. A number of intermediary services may be involved in the message path; a transport-level security technology will protect the privacy of the message contents while it is being transmitted between these intermediaries and number of additional technologies are required to facilitate the message-level security required for end-to end protection. WS-Security (Web Services Security Language) can be used to bridge gaps between disparate security models but also goes beyond traditional transport-level security to provide a standard end-to-end security model for SOAP messages through the use of header blocks that are sent with the message. Transport-level and message-level security mechanisms can be combined to obtain a higher level of security in service invocation.

In [8], security issues that occur in the underlying routing protocol are described, as well as fairness and trust issues that occur in file sharing and other P2P applications. The creation of a secure routing primitive is a technique that can be used to address this problem. In this model, application specific objects are assigned unique identifiers, called keys, selected from the same id space. Each key is mapped by the overlay to a unique live node, called the key's root. Such a primitive ensures that when a non-faulty node sends a message with a specified key, the message reaches all non-faulty members in the set of replica roots with a very high probability. Implementing the secure routing primitive requires the solution of three problems: securely assigning identifiers to nodes, to prevent an attacker from choosing the value of identifiers assigned to the nodes he controls; securely maintaining the routing tables, in order to ensure that the fraction of faulty nodes appearing in the routing tables of correct nodes does not exceed, on average, the fraction of faulty nodes in the entire overlay; finally securely forwarding messaging assures that at least one copy of a message sent to a key reaches each correct replica root for that key with high probability.

In [6] a general architecture is proposed that enhances security aspects by leveraging trusted computing technology, which is built on a trusted platform module and provides a mechanism for building trust into the application layer. A trusted reference monitor (TRM) in the platform of each peer can monitor and verify the information a peer provides and ensure data authenticity. Using credentials, a TRM can also digitally sign data from a peer and provide verification mechanisms for a remote site. The proposed general trusted platform with a TRM assumes an homogeneous environment, *i.e.* each platform is evenly equipped with the necessary trusted computing hardware.

Intrusion detection is another important issue in any security framework, in particular for mobile ad hoc networks, for which attacks on nodes can disrupt communications. Any node can observe part of the total traffic, making distributed intrusion detection a suitable approach: each node is responsible for detecting intrusions in its local neighborhood. In this form of intrusion detection, a node's neighbors observe that node's external behavior and form a judgement about a node' "trustworthiness". In [7] the common problem of how to combine observational data from multiple nodes is afforded with the Dempster-Shafer evidence theory. This theory addresses the problem of combining evidence from multiple observers representing uncertainty the form of belief functions. The idea is that an observer can obtain degrees of belief about a proposition from a related proposition's subjective probabilities. The Dempster-Shafer theory, compared for example with the Bayesian approach, is appealing because it can handle uncertainty or ignorance.

3 JXTA-SOAP

The JXTA-SOAP component is an official extension for the Java version of JXTA middleware, enabling Web Service sharing in peer-to-peer networks. JXTA-SOAP has been designed having in mind ubiquitous computing needs, to reduce the complexity otherwise required to build and deploy peer-to-peer service-oriented applications. In a previous work we described the internal architecture of JXTA-SOAP, with the purpose of conceptualizing its main features at a high abstraction level [13]. In particular we focused on service deployment, publication, lookup and invocation.

JXTA peers use pipes in order to exchange messages and access available resources. JXTA messages are XML documents with ordered message elements which may contain any type of payload. Messages are the basic data exchange unit in JXTA and all protocols are defined as a set of messages exchanged between peers. Pipes provide an asynchronous, unidirectional and unreliable communication channel by default. However, bidirectional reliable channels may be provided on top of them. They offer two operation methods: unicast pipes, which allow one-to-one communications, and propagation pipes, which allow one-to-many. JXTA pipes are an abstraction and are not bound to a specific IP address or port. They have a unique ID and are published just like any resource in the JXTA network, so any peer may update them whenever its physical location changes. Both input pipes, used for message reception, and output pipes, used for message sending, are considered pipe endpoints, an actual destination in the physical network. JXTA-SOAP uses pipes for carrying consumer-to-service requests, and service responses.

The internal architecture of a JXTA-SOAP peer is illustrated in figure 1.

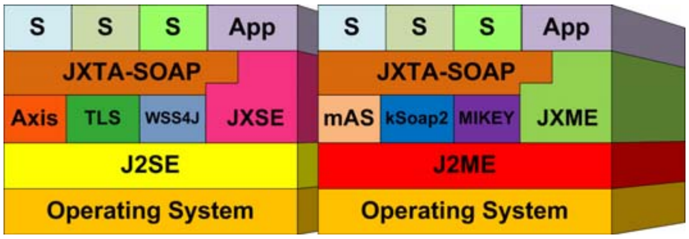


Fig. 1. Architectural layers of service-oriented peers based on JXTA-SOAP. The J2SE version (on the left) is based on Axis and JXSE, while the J2ME version (on the right) is based on kSoap2 and mAS.

We implemented two (interoperable) versions of JXTA-SOAP: J2SE-based, extending JXTA-J2SE, and J2ME-based, extending JXTA-J2ME. In the following we describe their features and the different technological solutions they rely on.

3.1 JXTA-SOAP for Java Standard Edition (J2SE)

The J2SE version of the JXTA-SOAP component supports service deployment, discovery, and invocation, with optional use of standard mechanisms to secure communications among peers. The core of the component is the Apache Axis engine (v1.4), which is a producer/consumer of SOAP messages. Usually Axis is deployed in a Web application server, such as Apache Tomcat, together with the implementations of the Web Services to be deployed, while client applications use the Axis Java API to create request instances. The Axis engine provides the processing logic, either client or server. When running, it invokes a series of Handlers according to a specific order which is determined by two factors - deployment configuration, and whether the engine is a client or a server. The object which is passed to each Handler invocation is a MessageContext, *i.e.* is a structure which contains several important parts: 1) a "request" message, 2) a "response" message, and 3) a bag of properties.

Once a service instance has been initialized, the Axis engine may dispatch multiple remote invocations to it. After that, when the Axis engine determines that the service instance needs to be removed from service of handling remote invocations, it destroys it. In the implementation of the destruction functionality, the service object releases its resources.

For remote service invocation, a consumer peer needs to instantiate a Call object. JXTA-SOAP's Call class extends Axis' default one, overloading the use of service URLs with the use of the Service Descriptor and the public pipe advertisement of the service. To create Call instances, the peer uses the implementation of the Call Factory class provided by Axis.

3.2 JXTA-SOAP for Java Micro Edition (J2ME)

The J2ME version of the architecture supports Connected Device Configuration (CDC) and Personal Profile. We implemented the API which enables the development of peers that are able to deploy, provide, discover and consume Web Services in a JXTA-SOAP network. Since Axis is not available for the CDC platform, we adopted kSoap2 [2] as SOAP parser (for consumer functionalities) and, for service provision, we integrated the mAS [3] lightweight engine.

Service invocation is allowed by a kSoap2 based implementation of the Call Factory class. The latter instantiates a kSoap2's Soap Object, and sets all the properties for message exchanging through JXTA pipes. Soap Object is a highly generic class which allows to build SOAP calls, by setting up a SOAP envelope. We have maintained the same structure of J2SE-based version for Call Factory, to allow portability of service consumer applications from desktop PCs or laptops to PDAs. Internally, the Call Factory class creates a Soap Object passing references to the Service Descriptor, the public pipe advertisement of the service and the peergroup as parameters for the creation of the Call object.

After instantiating the transport using the Call Factory class, the consumer peer creates the request object, indicating the name of the remote method to invoke and setting the input parameters as additional properties. This object is assigned to a Soap Serialization Envelope, as the outbound message for the soap call; Soap Serialization Envelope is a kSoap2 class that extends the basic Soap Envelope, providing support for the SOAP Serialization format specification and simple object serialization. The same class provides a getResponse method that extracts the parsed response from the wrapper object and returns it.

For service provision, we integrated the Server class of the Micro Application Server (mAS) into the basic service class of the JXTA-SOAP API. mAS implements the Chain of Responsibility pattern [4], the same used in Axis. It avoids coupling the sender of a request to its receiver by giving more than one object a chance to handle the request; receiving objects are chained and and the request passed along the chain until an object handles it. Moreover, mAS allows service invocation by users and service deployment by administrator; it also supplies browser management of requests, distinguishing if the HTTP message contains a Web page request or a SOAP envelope.

4 Secure Service Invocation

Currently, JXTA-SOAP supports secure service invocation by means of two orthogonal mechanisms. The first one, *transport-level security*, allows to create a TLS-based secure channel which guarantees the integrity and confidentiality of exchanged information, by means of mutual authentication between parties (using X.509 certificates) and data encoding. The other approach is WSS-based *message-level security*, for which SOAP messages sent by service consumers contain security parameters (tokens) which are extracted by service providers to check for consumers' compliance with the security policy of the invoked service. Figure 2 summarizes the interaction of a peer with a discovered secure service.

The `net.soap.jxta.security.policy` package provides two important modules, *i.e.* `Policy` and `PolicyManager`. The latter is a class which allows to associate a service with a security policy (currently: TLS, WSS, or both). The `Policy` interface provides methods which are commonly implemented by all policy classes. They allow to extract authentication parameters from the client request message and to verify their correctness according with the service security requirements. If all parameters are validated, the client is allowed to invoke the service, otherwise the request is refused. Authentication parameters are stored in a vector of `SOAPService` class that is used to keep a list of authenticated peers.

Current implementations of the `Policy` interface are `DefaultTLSPolicy`, based on `JXTAUnicastSecure` pipes which subsume default (unsecure) JXTAUnicast pipes, and DefaultWSSPolicy, which uses Apache's WSS4J to provide SOAP messages with security headers and fill them with tokens. Policies are associated to services by means of two extensions of the `<Parm>` field of the `ModuleSpecAdvertisement`. The first extension is the `<security>` tag, whose value (true or false) indicates whether a service is secured or not. The other extension is the `<InvocationCharter>`, a XML document which is inserted in the `<Parm>` field of the ModuleSpecAdvertisement if the `<security>` tag is set to true. The secure invocation model is illustrated in figure 3 (TLS-based case), with particular emphasis on multithreaded handling of concurrent invocations.

The `net.soap.jxta.security.certificate` package provides classes that enable JXTA-SOAP to use X.509 certificates for authentication and to extract them from a PeerAdvertisement; in particular, it uses the Key Store associated to JXTA PSEMembershipService, that is initialized when the peer boots in the network group. The `net.soap.jxta.security.wss4j` package implements a WSSecurity class, that enables the client to create a custom security header for invocation requests, according to the Invocation Charter (i.e. encryption, body signature).

4.1 DefaultTLSTransport

TLS is the default technology used by JXTA-SOAP when the secure service is created. Each SOAP message between client and server is sent through a TLS

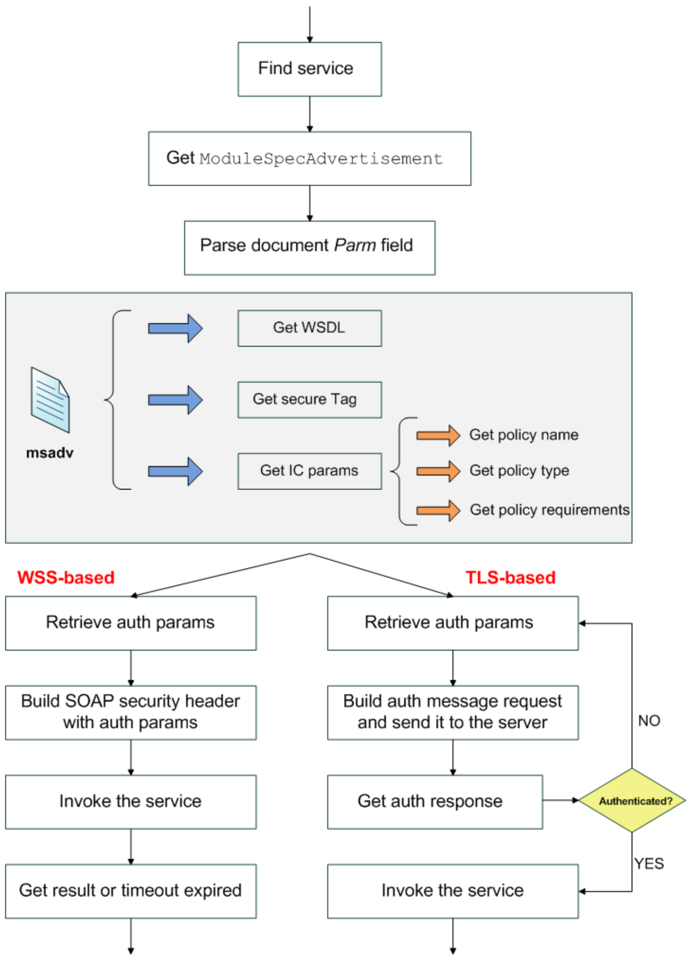


Fig. 2. Interaction with a discovered secure service

channel which requires previous authentication of involved entities. According to the implemented policy, a communication session is established, in which the client sends to the server its PeerAdvertisement with the peer self-signed certificate created by JXTA. The server is able to extract additional information about the requesting client (i.e. name, PeerID, group) and, if the authentication procedure succeeds, to update the list of authenticated peers with the corresponding entry:

[PeerID, X.509 Certificate]

Then a TTL (time to live) is associated to the authentication of the client. At the end of the period of validity, the entry is automatically deleted from the list.

Since the default TLS policy implementation uses JXTA's PSE keystore for storing/retrieving certificates, it is necessary that both client and server also

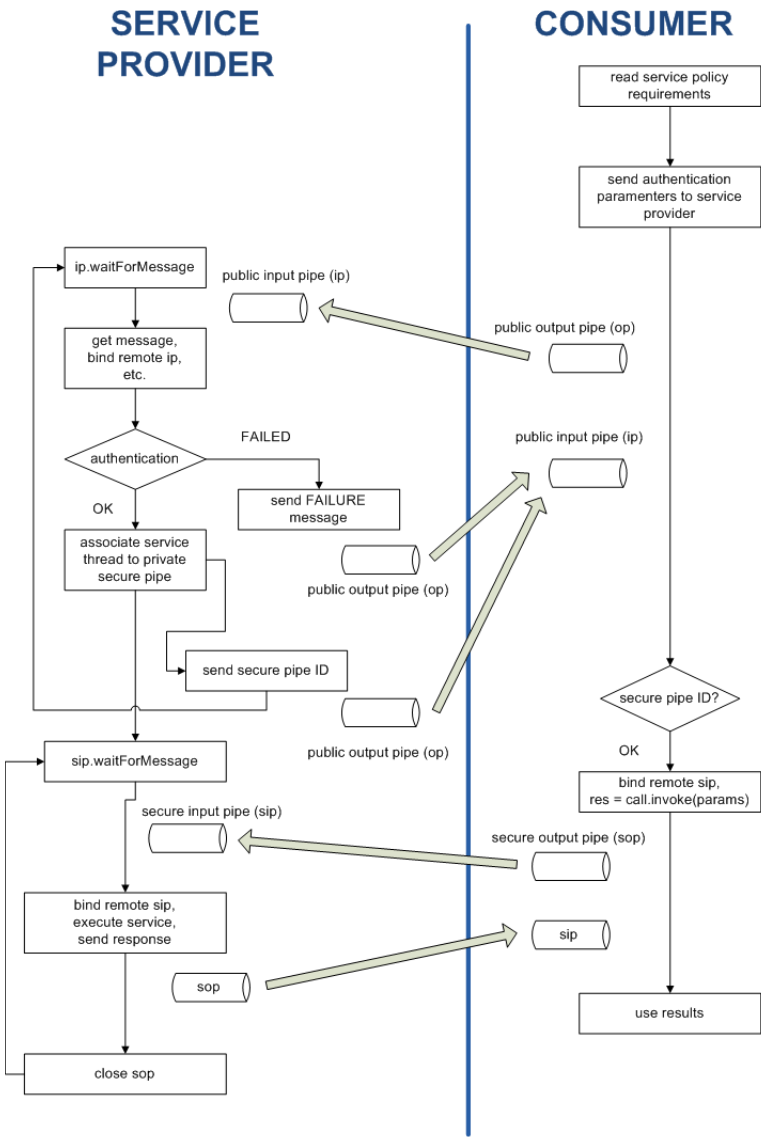


Fig. 3. TLS-based secure invocation model. Concurrent invocations are handled by separated service threads.

authenticate to the PSEMembershipService, which is the default membership service implemented in JXTA. When the client request is received by the server, an output pipe is created for invocation response and validation of authentication parameters; then the X.509 certificate is imported,a new thread is created for secure invocation management and the associated secure pipe is sent to the client for successive invocations.

4.2 DefaultWSSMessage

This is the default message security policy and uses the methods of WSSecurity class to build a security header which is included in all messages and invocation requests. During the authentication phase, the client attaches its X.509 certificate to the header using a `<wsse:BinarySecurityToken>` and digitally signs the request message body with its private key.

The WSS-based policy does not use JXTA PSE keystore, but requires that both client and server generate a couple of private/public keys and use them to create a self-signed X.509 certificate which is stored in their own keystore, whose integrity and privacy are granted by means of a password.

4.3 MIKEYPolicy for Mobile Applications

Since PSE membership classes are not available for JXME version of JXTA platform, we imported and modified them to be able to authenticate peers within a peer group. Moreover, J2ME does not support TLS, so it was impossible to use JXTA secure pipes for service invocation. We implemented a new type of pipe, `JxtaUnicastCrypto`, by which it is possible to cipher message contents, and we used them to define a new security policy, suitable for Connected Device Configuration (CDC) and Personal Profile.

`PipeService` and `PipeServiceImpl` classes in the `net.jxta.pipe` package create and use a SecretKey object containing the cipher algorithm and the corresponding key. The client and the server have to share the same key, so we introduced Multimedia Internet KEYing (MIKEY) [9] protocol to create the key pair and all the required parameters for encryption and decryption operations. Although memory and processing power have dramatically improved for handheld devices, encryption remains a resource-intensive task that requires consideration when designing protocols. MIKEY is a schema for management of cryptographic keys which can be used in real-time and peer to peer applications; it was developed with the intention to minimize latency when exchanging cryptographic keys between small interactive groups that reside in heterogeneous networks. The protocol is defined in RFC 3830 and in JXTA-SOAP project we introduced an implementation with RSA-R algorithm [10]. The standard describes mechanisms for negotiating keys between two or more parties who want to establish a secure channel of communication; to transport and exchange keying material, three methods are supported, Pre-shared secret key (PSK), Public Key encryption (PKE) and Diffie Hellmann (DH) key exchange. Each key-exchange mechanism (PSK, PKE, and Diffie-Hellman) defined in MIKEY is using the same approach of sending and receiving messages, but the message attributes (that is, headers, payloads, and values) differ from method to method.

To create a MIKEY message it is necessary to create an initial MIKEY message starting with the Common Header payload, and then concatenate necessary payloads of the message. As a last step create and concatenate the MAC/signature payload without the MAC/signature field filled in; calculate the MAC/signature over the entire MIKEY message, except the MAC/Signature field, and add the MAC/signature in the field. The common header payload must be included at the beginning of each MIKEY message (request and response) because it provides necessary information about the Crypto Session with which it is associated. MIKEY defines several payloads to support the three key exchange methods and the corresponding architectural scenarios (that is, peer to peer, simple one to many, many to many, without a centralized control unit). The main characteristic of MIKEY protocol is that it minimizes message exchange; the negotiation of key material should be accomplished in one round trip, as described in figure 4.

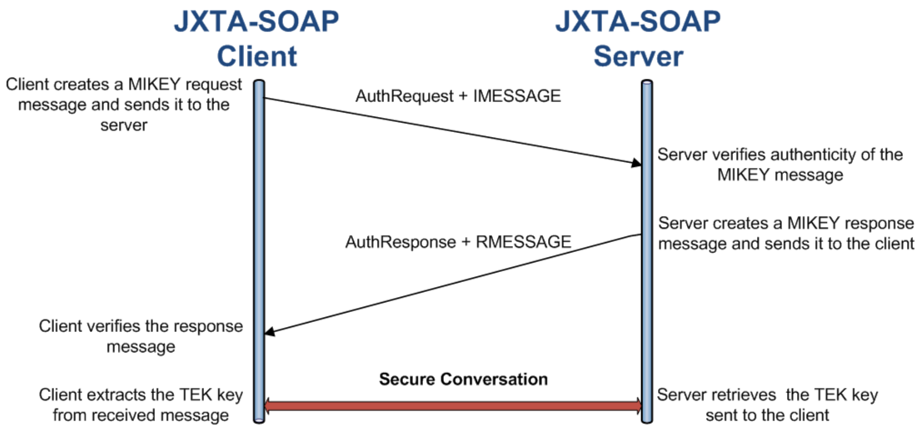

JXTA-SOAP Client
JXTA-SOAP Server
Client creates a MIKEY request message and sends it to the server
AuthRequest + IMESSAGE
Server verifies authenticity of the MIKEY message
Server creates a MIKEY response message and sends it to the client
AuthResponse + RMESSAGE
Client verifies the response message
Secure Conversation
Client extracts the TEK key from received message
Server retrieves the TEK key sent to the client

Fig. 4. MIKEY transaction example: the JXTA-SOAP Client is a peer that acts as service consumer, while the JXTA-SOAP Server is a peer that acts as service provider. Of course roles can be exchanged, since every peer can provide and consume services.

5 Conclusions

In this paper we presented the mechanisms that we implemented in the JXTA-SOAP component to support secure service invocation in a peer-to-peer network of service providers and consumers. In particular, we provided insights into the mobile version of JXTA-SOAP, targeting mobile devices with J2ME CDC profile.

Future work will mainly focus on supporting Web Service Security mechanisms also for the J2ME-based version of the component, in order to sign and verify SOAP Messages through the creation of security headers. This will ensure end-to-end security in service invocation when both server and client are resource constrained devices.

References

1. Amoretti, M., Bisi, M., Zanichelli, F., Conte, G.: Enabling Peer-to-Peer Web Service Architectures with JXTA-SOAP. In: IADIS International Conference e-Society 2008, Algarve, Portugal (April 2008)

2. Haustein, S., Seigel, J.: kSoap2 project, http://ksoap2.sourceforge.net
3. Plebani, P.: mAS project, https://sourceforge.net/projects/masproject
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Addison-Wesley, Reading (1995)
5. Govoni, D., Soto, J.C.: JXTA and security, JXTA: Java P2P Programming, pp. 251–282 (2002)
6. Zhang, X., Chen, S., Sandhu, R.: Enhancing Data Authenticity and Integrity in P2P Systems. IEEE Internet Computing (November-December 2005)
7. Chen, T.M., Venkataramanan, V.: Dempster-Shafer Theory for Intrusion Detection in Ad Hoc Networks. IEEE Internet Computing (November-December 2005)
8. Wallach Dan, S.: A Survey of Peer-to-Peer Security Issues. In: Okada, M., Pierce, B.C., Scedrov, A., Tokuda, H., Yonezawa, A. (eds.) ISSS 2002. LNCS, vol. 2609, pp. 42–57. Springer, Heidelberg (2003)
9. MIKEY: Multimedia Internet KEYing, http://www.ietf.org/rfc/rfc3830.txt
10. MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY), http://www.ietf.org/rfc/rfc4738.txt
11. Tang, J., Zhang, W., Xiao, W., Tang, D., Song, J.: Self-Organizing Service-Oriented Peer Communities. In: International Conference on Internet and Web Applications and Services (ICIW 2006), Guadeloupe, French Caribbean, February 2006, pp. 99–103 (2006)
12. Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J.-C., Poyoul, E., Yeager, B.: Project JXTA 2.0 Super-Peer Virtual Network, Technical Report, Sun Microsystems (2003)
13. Amoretti, M., Bisi, M., Zanichelli, F., Conte, G.: Enabling Peer-to-Peer Web Service Architectures with JXTA-SOAP. In: IADIS International Conference e-Society 2008, Algarve, Portugal (April 2008)
14. Erl, T.: Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall PTR, Englewood Cliffs (2004)

Security Aspects of Smart Cards vs. Embedded Security in Machine-to-Machine (M2M) Advanced Mobile Network Applications

Mike Meyerstein[*], Inhyok Cha, and Yogendra Shah

InterDigital Communications Corporation LLC,
King of Prussia, PA, USA
meyersmv@btinternet.com,
{Inhyok.Cha,Yogendra.Shah}@InterDigital.com

Abstract. The Third Generation Partnership Project (3GPP) standardisation group currently discusses advanced applications of mobile networks such as Machine-to-Machine (M2M) communication. Several security issues arise in these contexts which warrant a fresh look at mobile networks' security foundations, resting on smart cards. This paper contributes a security/efficiency analysis to this discussion and highlights the role of trusted platform technology to approach these issues.

Keywords: Smart card, UICC, machine-to-machine communication, embedded security, trusted environment.

1 Introduction

The idea of M2M is that un-manned terminals, e.g. traffic cameras, meters, cargo containers, can communicate with host servers using wireless global communications networks. This requires the usual secure authentication for network access.

The networks will not be specially M2M-enabled, so the authentication has to follow the standardised schemes currently in place for mobile (e.g. 3GPP) and fixed (e.g. WLAN) networks.

M2M security requirements [1] may make the conventional UICC (Universal Integrated Circuit Card) a less advantageous solution for secure authentication. It is necessary to look at the options for a non-personalised security module to which a network operator's MCIMs (Machine Communications Identity Modules) can be downloaded [1]. This may be accomplished using an embedded Trusted Environment (TRE) in a terminal. The TRE acts as a hardware root of trust for the storage and execution of secure applications and may also have protected software functions. The TRE may host downloaded software MCIMs that emulate the behavior of the USIM (Universal Subscriber Identity Module) [2] or ISIM (Internet Multimedia Services Identity Module) [3] applications.

[*] Mike Meyerstein is the proprietor of Meyerstein Consulting Ltd, currently providing consultancy services to InterDigital Communications Corporation.

2 M2M Requirements

The M2M market has some definitive characteristics [1]:

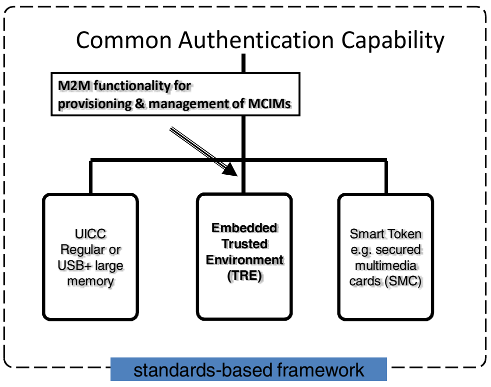
- Terminals may be in hard-to-reach locations (e.g. traffic cameras)
- Terminals may become geographically dispersed over time (e.g. cargo containers)
- Owners of large populations of terminals may want to change the network operator without visiting the terminals (e.g. to change the UICC).
- Terminals need to be protected against unauthorised removal of UICC
- Terminals may require over-network provisioning after sale or installation.

3 The Options for a TRE to Host Secure, Downloadable MCIMs

Client-side technologies for TREs could include

- UICC with download capability.
- An embedded TRE in the terminal, to provide a secure execution and storage environment. MCIMs would be downloaded to the TRE over public IP networks.
- Smart token such as the new multimedia card with on-card UICC (or "SMC" – Secure Multimedia memory Card)

A framework of standardised specifications is needed for the above solutions.



Common Authentication Capability
M2M functionality for provisioning & management of MCIMs
UICC Regular or USB+ large memory
Embedded Trusted Environment (TRE)
Smart Token e.g. secured multimedia cards (SMC)
standards-based framework

Fig. 1. Options for TRE to Host Downloaded MCIMs

The discussion within standardisation about the (dis-)advantages of the various candidate solutions is lively and far from concluded.

In Table 1 on the next page we collect the main arguments that have been advanced thus far in various forums and standardization committees (there is no reference document in which this information can be found).

Table 1. Comparison of Potential Solutions for TRE

REQUIRED FEATURES	Today's UICC	Embedded TRE	SMC
Currently standardised	good	medium (could be based partly on TCG specs)	medium
Currently available	good	medium (some limited versions available)	poor
Protection against unauthorised removal	poor (good, if M2M form factor UICC is soldered in)	good	poor
Provides secure API	good	good	not known
Does not require connector and interface chips	poor	good	poor
MCIMs can be downloaded for initial provisioning and for replacement of MIDs	poor. Can only install USIM or ISIM during manufacture	Good	poor. Can only install USIM or ISIM during manufacture
Key management suits M2M model	poor (pre-shared keys for authentication and download)	good (can use PKI (Public Key Infrastructure))	poor
Predictable costs	poor (for full-function, big memory, downloadable UICC)	Good (part of chipset)	poor
Secure channel to terminal	poor (standardised but seldom implemented)	good	not known
Remote change of operator	poor	good	poor
Open API for download	poor	good	poor
Track record of trust	good	poor	poor
Established infrastructure	good	Poor	Good (fits UICC infrastructure)
Built into operator's current trust models	good	Poor	Medium (partly fits UICC trust model)
Built into operator's current business models	good	poor	poor
Promotes operator's current brand image	good	poor	good

No solution comes out as perfect, but the embedded TRE shows promise if it can be standardised. The UICC shows promise if its current limitations (including that of lack of implementation of already-standardised features in cards and terminals) can be overcome. In the next sections, we look at issues surrounding the use of UICC as a TRE.

4 Smart Card Security in Mobile Networks: Why Is the Smart Card a Trusted Anchor?

Physical Tamper-Resistance:
An ISO -7816 [7], [8], [9] smart card has, in practice, a single-chip architecture with little possibility of monitoring communications between different chips on the card. Smart card ICs (Integrated Circuits) are designed and implemented to prevent probing and reverse-engineering. They are fabricated on a dedicated production line in a secure facility. Measures include scrambling of busses and of memory addresses, bonded passivation layers, permanently disabled test points, self-generated programming voltage.

Proprietary, Secure O/S (Operating System):
The Smart card's standardised API, e.g. [4], consists of a restricted command set that has no hidden commands or access methods. It is trusted because of its own built-in security mechanisms and because of those of the underlying hardware platform. It is non-updateable.

For applications such as USIM, the K (i.e. the pre-shared authentication key)and OTA (Over The Air) keys are stored and accessed by the O/S in proprietary ways. The O/S cannot be made to reveal the values or memory locations of those data.

Conventional GSM SIM (Subscriber Identity Module) cards did not allow adding applications to a live card. The advent of the Javacard [10] now allows applications on a multi-application UICC platform to be updated, deleted or added to an issued card, either remotely or locally. The potential of Javacard for Network Operators is currently restricted by

- Implementation by Network Operators of only SMS (Short Messaging Service) as the bearer (for OTA messages to the UICC), which has a very limited bandwidth
- OTA security standards [11][1] are not profiled for IP bearers
- Lack of a sufficiently rich terminal/UICC interface on nearly all MEs[2] (Mobile Equipments)
- General concerns about the security of multi-application Javacards.

In the world of telecoms, it is generally the buyer's task to perform due diligence tests on the smart card vendor to ensure that the O/S has been properly developed and evaluated.

Other Measures:
Smart cards include proprietary measures to prevent attacks such as slowing down the external clock and measures against power analysis attacks by the use of noise-free

[1] N.B. [11] has been recently split up and its former contents have been dispersed over [18], [19], TS 31.115 Secured Packet Structure for (U)SIM Toolkit applications and TS 31.116 Remote APDU Structure for (U)SIM Toolkit applications. [11] is still widely referred to in the telecoms sector.

[2] A few terminals have implemented the JSR177 [12] terminal/UICC interface, but it's usually only the SIM toolkit part and not the general APDU (Application Protocols Data Unit) API. A few Windows Mobile MEs have allowed an open APDU API to the UICC, using the terminal's RIL (Radio Interface Layer) but it is not clear if those are still in production.

computational algorithms and/or injection of artificial noise and/or damping of noise on the power rail. There are also said to be a large number of detailed precautionary measures taken, some of which are described in the public domain (e.g. [14]).

Design and Development Process:
Security is designed into a smart card IC in the secure facilities of a semiconductor manufacturer. The computers that are used for this are isolated from the rest of the world. Undocumented counter-measures in the IC are supported by corresponding design criteria. Once the O/S development is finished, the entire source code may be checked by an independent evaluation.

Supply Chain:
There are only a handful of world-class vendors of smart cards, so it is feasible for Network Operators (who own and specify the UICCs) to perform the necessary security audits. There will be an agreed arrangement for transferring the K objects between network operators and UICC vendors. K values cannot be retained in the vendors' personalisation systems or be discovered by system operatives.

Security Evaluations:
Card vendors have their O/S independently evaluated and MNOs perform security evaluations of their card vendors' products and facilities. GSMA (GSM association) [15] provides non-public guidance to its members on how to do that. Common Criteria Protection Profiles have been published for smart cards. One of these [16] is aimed at the underlying IC platform but some are aimed at payment cards issued by financial institutions such as Visa and Mastercard. Cost can be an issue for wider adoption of these evaluation regimes. There are no standard specifications or protection profiles for the security evaluation of telecoms smart cards such as UICCs.

UICC-Terminal Interface Security:
The UICC employs some security measures in the interface with the terminal:

- User Authentication: PINs (Personal Identity Numbers) (called CHV1 (Card Holder Verification) and CHV2 in a SIM card), provide some level of protection with user authorisation on the interface. CHV1 can be disabled by the user, in which case there is no PIN-protection for making calls. The use of CHV1 and CHV2 poses a security vulnerability since the passwords get transported across the UICC-ME interface in clear. The UICC does not authenticate itself to the user, although 3G authentication [2] provides mutual authentication of the card and the network. In M2M, a remote user could rely on two possible methods of assuring himself of the authenticity of a UICC, i.e. (a) using a remote access protocol that exploits a pre-shared or private key on the UICC and/or (b) using e.g. Liberty Alliance [22] protocols to trigger the UICC issuer to perform an authentication of the UICC and possibly binding that to the remote access session,
- Secure channels: Commands to the UICC are not secured unless they are inside a 3GPP OTA envelope [11]. That is why ETSI and 3GPP have recently specified secure channels and their key establishment methods ([5], [6]) across the terminal-UICC interface. ISO (International Organisation for

Standardisation) 7816 [20], [21] and EN726 [17] define secure messaging between a terminal and a smart card but key distribution was not defined (it being assumed that it would be based on pre-installed keys). ISO7816 also defines a set of security-related commands. Neither the ISO nor CEN (Comité Européen de Normalisation) techniques are included in UICC specifications such as [4]. ETSI (European Telecommunications Standards Institute) and 3GPP secure channel specs [5] and [6] compliment the ISO and CEN standards by defining methods for key distribution between the terminal and UICC. There does not seem to be any reason to believe that normal terminals can be trusted to store the distributed keys. Protocols such as Global Platform [10], ETSI RAM (Remote Application Management)/ RFM (Remote File Management) [18], [19] and 3GPP OTA [11] provide end-to-end security from server to card for the purpose of loading and managing files and applications on the UICC. They do not require a secure ME/UICC interface.[3]

- Protection of data across the interface: All standardised command-sets of a UICC are designed to be sent in the clear across the terminal/UICC interface. Protocols that may be subject to replay attacks must have counter-measures built in, e.g. the sequence numbers used in 3G authentication [2]. In future, the Smart Card Web Server (SCWS) [23], could use HTTPS (HyperText Transfer Protocol Secure)/TLS (Transport Layer Security) to establish a secure tunnel from card to server (or to terminal) via which usernames and passwords could be sent.
- Access control: In general, ACLs (access control lists) are not used in today's UICC O/Ss. Access to file operations relies on the principle that if the entity accessing the file can satisfy the access policy (embodied in the File Control Parameters [4]), then it must be an authorised entity.

5 Meeting M2M Requirements with UICCs

Advent of the "Big SIM" UICC

Recent innovations in smart card technology could go a long way to enabling the UICC to fulfill the M2M requirements. The new features described below, plus the ability to store downloaded applications and multimedia files, would need the large memory of "Big SIM". However, providing such features on UICCs could well be cost-prohibitive for M2M.

Large Memory: Recently, UICCs with flash memory of up to 2Gbytes have become available.

[3] Remote Application Management can theoretically be used to download any Javacard application to a UICC and to store it in a security domain. It does not currently apply to the U(I)SIM applications, as there is no standardised mechanism for the UICC to extract and store the Ki and algorithm customisation parameters. For the case of updating existing files either locally or remotely, this is possible only where the access conditions in the File Control Parameters can be satisfied. Remote (OTA) file update is possible on files in any application on the UICC, but only if the files were OTA-enabled at the time the file was created on the UICC.

High-Speed I/O: The conventional I/O speed of the UICC/terminal interface is only a half-duplex 9.6Kbit/s[4]. In order to be able to move data on and off the Big SIM in a meaningful timeframe, a USB (Universal Serial Bus) interface has been specified in [13] and [8].

Smart Card Web Server (SCWS): The advent of "Big SIM" with USB I/O enables the UICC to support an IP stack and web server. There are a number of advantages to this, e.g. use of (X)HTML ((eXtensible or) HyperText Markup Language) to communicate with the UICC. UICCs could even have their own IP addresses, which could introduce a whole set of security issues. ETSI SCP has standardised SCWS [23] and IP [23] on a UICC. Use of SMS for application download is limited in practice to about a 1kByte payload, i.e. 7 concatenated SMSs. Even then, this requires a dedicated SMS-C (SMS Centre) to achieve an effective success-rate. With ordinary SMS-Cs whose resources are shared with mainstream SMS, the practical limit may be as little as 2 concatenated SMSs, i.e. about 300 bytes. The size of a download using an IP bearer does not suffer from such limitations.

Internal Security Domains: Global Platform has specifications [10] that define security domains on a Javacard smart card. ETSI SCP (Smart Card Platform) are now expanding upon these in their specifications for "Confidential Applications." This allows the card issuer to set up domains for the use of third parties to load applets onto the card. The issuer cannot examine those applets. The UICC provides a sandbox environment in which the domains are isolated from each other – a feature that has been somewhat limited in Javacard implementations up to now.

Enhancements Required to UICCs for M2M Mass Market

A UICC to be used in mass-market M2M applications would have to support the requirements of long life-time with long maintenance intervals, non-removability, remote download of operator's authentication application and remote change of operator. Some very significant enhancements need to be considered as follows:

Security Domains: Support is required for security domains for the card issuer and for third parties, e.g. as per the SCP specifications "Confidential Applications" concept. But in the M2M scenario, network operators would be classed as third parties. In order for the card issuer to allow the M2M equipment owner to change to a new network operator, the card issuer (who is therefore not a network operator) assigns a domain to a new network operator and closes the domain of the old network operator.

Removable UICC vs. Downloadable UICC: unauthorised removal of a traditional "removable" UICC must be made very difficult, while its replacement must be easy. A better alternative to this is to fix the UICC in the terminal and for it to support the ability to download MIDs. Such a UICC must be able to extract K objects and similarly sensitive data from messages from a remote server and lock them away in secure memory so that they cannot be revealed to entities outside the UICC. Network operators will demand that there be no reduction of security in UICCs that support these features.

Download Protocols: Support is required for secure download protocols other than standardised OTA, i.e. M2M requires protocols which do not require pre-shared keys

[4] Somewhat higher speeds can be negotiated between ME and card, but all must support the default speed for backward compatibility.

and which can be used over IP bearers. (In this respect, support for SCWS and IP stack could be an advantage).

Secure Interface to UICC: Support for a secure terminal-UICC interface [5], [6] may be a requirement, as discussed above.

6 Security Analysis and Comparison: Can an Embedded TRE Ever Be as Secure as a Smart Card?

Successful, Publicised Attacks Against Smart Cards
Smart Card technology had experienced some crises in the past, as follows:

Side Channel Attacks (specifically Power Analysis): This type of attack relies on the noise on the smart card's power contact being correlate-able with the processing that is going on in the UICC, especially when it is reading the values of a secret key into its internal registers. This can work well on an unprotected card that uses the DES (Data Encryption Standard) algorithm. RSA (Rivest, Shamir, Adleman) is not susceptible and neither is AES (Advanced Encryption Standard) (upon which 3GPP Milenage is based).[5] It is easy to prevent this type of attack at the design stage of a smart card. Likewise, an embedded TRE would be designed and implemented such that it would not leak this information. There would be no such interface that an attacker could monitor and the processing in the TRE would be such that any noise occurring on any power rail would not contain any recoverable information.

Probing of Broadcast TV Cards: Satellite TV smart cards have to contain global decryption keys, due to the nature of the service, i.e. a broadcast encrypted signal. If you crack one card, you have cracked the whole scheme. This is, of course, not true of networked telecoms systems. In the early 1990s, Cambridge University in the UK used physical probing to successfully recover secret keys from satellite TV cards. It is not necessary for embedded TREs (or UICCs, for that matter) to contain global keys and their embedded nature would make physical probing infeasible.

Cloning of SIM cards: The only verifiable instance of cloning of 2G SIM cards was a "known plaintext" (50,000 challenge-response pairs) attack against the COMP128 authentication algorithm and not against the card platform itself. This attack has been prevented by the use of better algorithms and is not at all possible with 3G. The relevance of this attack to TREs is that it has gone down in the annals of urban mythology as an attack on the SIM card, rather than an attack on the algorithm. "Mud sticks," as they say.

Attacks Against Disposable "Eurochip" Phonecards: Two successful attacks exploited (1) a weaknesses in the card's hard-wired logic and (2) the absence of a security module in some payphones. The relevance of this to embedded TRE is that many attackers seem able to acquire insider knowledge of the product. "Security by Obscurity" is not a viable counter-measure.

Lack of Security Specifications and Formal Evaluations for UICC: Tamper-Resistance: There are no standardised specifications as such that assure the degree of

[5] This attack was successfully perpetrated against the pilot of a well publicised smart card payment system in the UK in the early 1990s. Although the scheme was designed to use RSA with unique private keys, the pilot used DES with a global key on every card.

physical and logical tamper-resistance described above. It is up to the buyer to specify what he wants, although, in theory, protection profiles such as [16] could be adapted for UICCs and evaluations enforced by buyers. The same might also be true of embedded TREs, unless they use TCG-style technology and also conform to suitably-created and enforced protection profiles (which might need to be wider in scope than those specified so far by TCG for Trusted Modules (see http://www.bsi.de/zertifiz/zert/reporte/pp0030b.pdf).

Perceptions vs Real-World Implementation of UIC**C:** There is never a guarantee that, for a given advertised UICC product, all or any of the possible counter-measures have been implemented. It is a case of being an informed buyer. Network Operators can obtain un-published information about counter-measures from card vendors and large-volume buyers can specify their own counter-measures, within the constraints of the silicon manufacturing process. This should be borne in mind when some commentators argue that an embedded TRE cannot be as secure as a UICC.

Secure Terminal/UICC Channel**:** Even if the ETSI/3GPP specifications concerning key establishment [5], [6] are implemented, it is not specified how the terminal securely stores the local key and executes the algorithms. There have been failed attempts in the past (e.g. the MET – Mobile Electronic Transactions forum) to portray the terminal as a PTD (Personal Trusted Device). It would be fair to say that the telecoms industry has little confidence in the ability of terminal suppliers to implement a secure environment.

Arguments Against Trusting a Non-UICC TRE

Some network operators may have the following perceptions, as expressed in some draft versions of [1], about a non-UICC TRE in a terminal. Counter-arguments are in italics.

- Network operators can trust the UICC because they are in charge of the specifications and they buy them directly from their approved vendors. They will not be able to do this for TREs. *It will be necessary to have an international accreditation scheme for TREs.*
- The telecoms industry has little confidence in the ability of terminal suppliers to implement a secure environment (see above). This is not helped by the suppliers who are involved in standardising M2M but who keep trying to limit the functionality of the TRE. *Once again, a security accreditation scheme is needed.*
- Compared to the small number of UICC vendors, there are potentially many suppliers of M2M terminals and therefore of embedded TRE solutions. *As is the case with UICCs and secure semiconductor products, the number of companies that decide to position themselves as TRE manufacturers could turn out to be limited.*
- The UICC is a product with a great track record. The O/Ss have been evolved over many years. A non-UICC TRE is still at the conceptual stage. *No counter argument except that every technology is eventually superseded by progress. UICCs were new, once upon a time.*
- Network operators want to use OTA infrastructure for downloading MIDs. Current OTA is not secure enough and would need to be updated, but that

could cause backward-compatibility problems. Other protocols, which could be used over an IP bearer, could well be secure enough but would have to be evaluated. *M2M equipment vendors will have no problem in providing the required APIs to an embedded TRE to allow use of an IP bearer and appropriate download protocols. Network Operators could possibly be persuaded, since some of them were very active in Liberty Alliance [22]. Also, IP download could be sub-contracted.*

- Today's system for distributing K from UICC manufacturers to Network Operators is well understood and well tried. Network Operators take the view that the key distribution required in the proposed Candidate Solution 1 in [1] would involve passing the K around an increased number of entities, which could introduce vulnerabilities. Scale-ability of proposed solutions in [1] is also an issue. *The number of parties involved with the key distribution does not have to increase, since the required functions can be combined into a small number of real-world roles. Also, K could be end-to-end encrypted (with the TRE's public key) by the entity that generates it. The system solution for embedded TRE in [1] is intended to be scale-able, requiring normal roaming agreements for initial connectivity and standardized, PKI-based protocols for TRE validation/authentication and MCIM download.*

7 Conclusions

An embedded TRE, if properly specified, implemented and certified, could provide a much-needed alternative to the UICC in meeting the requirements of M2M.

An embedded TRE challenges the long-established infrastructure and practices of Mobile Network Operators and UICC suppliers, but it could be a key enabler for the potentially huge M2M market to take off, thereby becoming an important new source of revenue.

If an embedded TRE is to be a competitor to the UICC for M2M, it must have sufficient resistance to the threats that are described in the threat analysis in [1]. Ongoing co-operation between TCG and standardization bodes involved with communications aspects of M2M could produce appropriate security specifications.

Whether or not the previously successful attacks described above could be used to perpetrate such threats depends on the implementation of the embedded TRE, e.g. whether it is a discrete component, part of a single-chip CPU (Central Processing Unit), system ASIC (Application-Specific IC), or distributed across a chipset. It is not possible to provide an implementation-independent answer to the question: "How difficult would it be for an embedded TRE to have some of these measures?" because some of the measures might not be relevant. An ASIC makes things easier in terms of security evaluations and accreditations.

Specifiers, developers and implementers of embedded TREs must bear in mind the lessons which a long experience of smart cards has taught us:

- Side-channel attacks would be much more difficult to achieve against a TRE if the TRE is not a discrete component
- One can argue that direct probing of a TRE is not important, as an individual TRE does not contain any global secrets. Nevertheless, there are always

people who will take up the challenge of attacking the TRE, for example, in an attempt to extract a K. Such threats are faced by every security initiative and only provide further motivation to design it correctly.

- Attacks against the cryptographic algorithms in a TRE could be publicised as attacks against the TRE.
- Many attackers seem able to acquire insider knowledge of the product. “Security by Obscurity” is not a reliable counter-measure. Do not rely on attackers being unable to gather information on how a TRE works.
- Reputation is important. The UICC vendors have enormous reputations in the world of security, whereas the terminal suppliers have little or none. TREs should be manufactured and personalised by reputable chip manufacturers with a proven track record in the security industry.
- Perceptions are important. The perception of the UICC may, in some cases, exceed its actual specification or design. Such must not be allowed to become the case for TRE.

References

1. 3GPP TSG WG3 unapproved draft technical report TR33.812 (current version 8.0.0) Feasibility Study on Remote Management of USIM Application on M2M Equipment (this is a working title which can change at any time)
2. 3GPP TS 31.102; Characteristics of the USIM Application
3. 3GPP TS 31.103; Characteristics of the ISIM Application
4. ETSI TS 102 221: UICC-Terminal interface; Physical and logical characteristics
5. ETSI TS 102 484 Smart Cards; Secure Channel between a UICC and an end-point Terminal
6. ETSI TS 33.110 Key establishment between a UICC and a terminal
7. ISO 7816-1 Identification cards – Integrated Circuit Cards - physical characteristics
8. ISO 7816-2 Identification cards – Integrated Circuit Cards - dimensions and location of contacts. AMD1= assignment of C4 and C8 (2004)
9. ISO 7816-3 Identification cards – Integrated Circuit Cards - electrical interface & Tx protocols
10. Global Platform specifications, v 2.2, may be, downloaded from `http://www.globalplatform.org`
11. 3GPP TS 23.048 Security Mechanisms for SIM Toolkit Application; Stage 2. N.B. this has been recently split up and its former contents have been dispersed over [18], [19], TS 31.115 Secured Packet Structure for (U)SIM Toolkit applications and TS 31.116 Remote APDU Structure for (U)SIM Toolkit applications. TS23.048 is still widely referred to in the telecoms sector
12. Java community specification JSR177: Security And Trust Services API
13. ETSI TS 102 600: Characteristics of the USB Interface
14. Rankl, Effing: The Smart Card Handbook, 3rd edn. Wiley and Sons, Chichester
15. See, `http://www.gsmworld.com`
16. Eurosmart: Smart Card IC Protection Profile, PP-0002, first published (2001) (EAL4 augmented)
17. CEN standard EN726 Identification card systems. Telecommunications. Integrated circuit(s) cards and terminals. There are 7 parts to this standard

18. ETSI TS 102 225: Secured packet structure for UICC based applications
19. ETSI TS 102 226: Remote APDU structure for UICC based applications
20. ISO 7816-4 Identification cards – Integrated Circuit Cards -Organisation, security and commands for interchange
21. ISO 7816-8 Identification cards – Integrated Circuit Cards - Commands for security operations
22. Current specifications for ID-FF and ID-WSF, `http://www.projectliberty.org`
23. TS 102 483: UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal

Simple Peer-to-Peer SIP Privacy

Joakim Koskela[1] and Sasu Tarkoma[2]

[1] Helsinki Institute for Information Technology, P.O. Box 9800,
FI-02015 TKK, Finland
joakim.koskela@hiit.fi
[2] Computer Science and Engineering Department,
Helsinki University of Technology
sasu.tarkoma@hut.fi

Abstract. In this paper, we introduce a model for enhancing privacy in peer-to-peer communication systems. The model is based on data obfuscation, preventing intermediate nodes from tracking calls, while still utilizing the shared resources of the peer network. This increases security when moving between untrusted, limited and ad-hoc networks, when the user is forced to rely on peer-to-peer schemes. The model is evaluated using a Host Identity Protocol-based prototype on mobile devices, and is found to provide good privacy, especially when combined with a source address hiding scheme. The contribution of this paper is to present the model and results obtained from its use, including usability considerations.

Keywords: peer-to-peer, privacy, P2PSIP, HIP.

1 Introduction

Privacy in peer-to-peer (P2P) systems has been an active topic for a number of years, having touched upon a wide range of aspects of this area of computing. Preventing identities or actions from being disclosed to others is difficult in P2P systems, which per-definition rely on the close cooperation between nodes.

Much of the related work has concentrated on hiding the real identities of users [1] [2] [3], as this has been the most pressing issue partly due to the legally questionable use of many P2P content-sharing systems. As long as the users cannot be traced, there is not a need to hide the actions (content requests) unless the content itself reveals something important.

Internet telephony, such as Skype, has traditionally relied on a system of trusted, centralized, servers for authentication and setting up connections. The privacy of the users, with respect to the system operator, is non-existent. Although phony accounts, source address hiding and voice scrambling can be used to protect end-users, the operator has all the means to track calls made through the system. This may not be a concern for most consumers, since they can be assumed to trust the operator; however, for companies and governments this raises more fundamental issues. For example, many companies block Skype, because call routing is proprietary and there is no way to guarantee privacy and confidentiality of the calls.

As we move toward mobile and ad-hoc environments, we need fully distributed systems that operate without the help of centralized nodes. This has resulted in initiatives such as the IETF's P2PSIP working group [4], aimed at standardizing a fully distributed session signaling protocol. In these systems, the control is shared amongst the peers, with the implication that anyone that is part of the network can track the calls made through it. As the operations themselves (such as call setup) implicate the users involved, simply hiding the source of these requests does little. Indeed, a more systematic privacy mechanism is needed.

Although several proposals have addressed this problem domain in related work, implementation results have not typically been elaborated. In this paper, we examine a simple application-level model for enhancing privacy in fully distributed communication systems. First, we review the details of the threat in current systems. We follow this by presenting our solution, discussing its benefits, tradeoffs and possible alterations. As the model imposes restrictions on the accessibility of users, we continue by discussing issues related to the usability of the solution. Finally we present our implementation and results obtained from its use.

The new contribution of this paper is to present and evaluate, through a prototype implementation, a concrete model for enhancing privacy in P2P communication systems. Also, we highlight and analyze issues concerning the usability of the model and present our solution for these. We hope that this paper would foster discussions of how privacy is perceived in P2P environments, possibly leading to new opportunities for further research.

2 Problem Scope

Peer-to-peer systems form a network between the participating nodes, used to collectively perform tasks and manage resources. In P2P communication systems, this network is used to perform the duties of a service provider. Most importantly to establish the voice or other calls, but also for managing presence, contact lists or for other services. A common model is to use the peer network as a distributed storage and message routing overlay, where the information needed to contact users is stored, scattered throughout the nodes of the network.

This is also the approach taken by the IETF P2PSIP working group [4], arguably the most prominent attempt at creating an open standard for these systems. The name is derived from peer-to-peer Session Initiation Protocol (SIP), as the original intent was to create a P2P version of the successful multimedia session signaling protocol.

The current version of the P2PSIP protocol draft defines a highly modular framework supporting different applications (called *Usage*s) as well as overlay network types [5]. The network module of this framework offers message routing, key-based storage and connectivity services through a common interface, independent of the underlying network structure. Although there are few restrictions, the underlying network is assumed to be a distributed hash table (DHT)-like structured network, with efficient key-based routing, where the storage service scatters the data throughout the network.

SIP-based applications use this framework for three primary operations, referred to as the *SIP Usage*; registration, lookup and connection establishment. Registering a SIP identity (SIP Address of Record, SIP AOR) with the overlay is done by creating a data packet containing the SIP AOR and a NodeId, the identifiers used for nodes in the overlay, or another SIP AOR through which the users can be reached. This packet is stored in the network under a key made from the hash of the SIP AOR.

The session signaling is exchanged directly between peers. To establishing this connection, peers use the *Attach* function provided by the network module. This initiates an Interactive Connectivity Establishment (ICE) [6] procedure, during which *address candidates* are gathered, and sent to the other peer through the overlay network. These address candidates are all Internet protocol (IP) address and transport-level port combinations through which the peer might be reachable (due to multihoming, network address translation or firewalls). These three operations offer a number of opportunities for curious, or malicious, users to eavesdrop.

Assuming a DHT-like overlay (such as Chord [7]), where responsibility for data is assigned using a key proximity function, any node with a suitable NodeId, along with every node in-between, is able to intercept registrations. Even if these nodes were trustworthy, it is trivial to request the information using the public SIP AOR.

To establishing a SIP session, the caller fetches the registration packet for the responder, and connects using the Attach function. Again, the node maintaining the registration packet, and all nodes in-between, can easily monitor from whom the user receives calls. And even though that path is secured, Attach-related messaging offers yet another opportunity to track the call.

As noted in the draft [5], end-to-end encryption of the payload could be used to mitigate some of these, although still leaving the storage keys and NodeIds exposed. Also, a strong authentication mechanism and a protective identity acquirement procedure would prevent peers from situating themselves suitably in the network, decreasing the possibility for eavesdropping and sabotage in large networks [8].

The problem at hand is to introduce privacy mechanisms to P2PSIP that guarantee end-users that their sessions are not intercepted and tapped by possible malicious peers. The threat is anticipated by the P2PSIP Internet Draft; however, to our understanding this paper presents the first analysis of P2PSIP privacy issues based on experimentation.

3 Solution Model

In the design of the privacy-enhancement for P2PSIP, we identified the overlay data storage as a crucial component. The aim was to design a simple model which sets no additional requirements on the underlying storage service, using only the get, put and remove primitives. This way the model can be adopted to other, similar systems as well. Besides, this does not prevent the use of additional privacy-related enhancements, such as source address hiding, for even greater security.

We did make the assumption of a strong, cryptography-based, identity scheme. The identities are associated with the public key of a key pair, with the private key kept secret by the rightful owner. This is a reasonable assumption, as it reflects the current trend in these systems and is a common solution for identity management in distributed systems in general. It allows users to protect the integrity of their data and make it accessible only for specific peers using encryption. How these keys are generated or bound to the identities, e.g., using a trusted third party certificates or leap-of-faith, is not relevant for the privacy scheme.

This leads us to the first step of our solution, which is to encrypt the registration packet using the public key of peers that may want to establish a session. To protect the integrity, the packet is signed prior to encryption, preventing either the data, source or intended recipients from being revealed without the proper decryption key. We thus *publish* this data only for a specific set of trusted peers. As each peer uses a different key pair, we need to encrypt and publish the package multiple times or use a common encryption key appended as a set of attachments, each encrypted using a different public key.

3.1 Storage Key Obfuscation

As the keys used for storing registration packets map directly to the public identities, encrypting the content of these packets alone does not prevent intermediate peers from tracking the calls. Size analysis can be used identify registration packets and by monitoring the keys used to store and request these, we can determine both who is on-line as well as the recipient of a call. Although buffering and *decoy* packets can be used to make it harder, it seems unavoidable that the storage keys should also be obfuscated.

Until now, we have assumed that users need to possess only the public keys of the trusted peers to publish registration packets. We could consider a scheme were the storage key is also encrypted using these. However, this is easily broken in systems were the public keys are well known, which is our assumption. An attacker could not make a general query whether a specific user is on-line, but being aware of even one peer which that user trusts, be able to query the state of the user with regard to that peer.

We could also consider a scheme where all users publish their registration packets (encrypted) for a specific peer using a common storage key. Either as such, or under a secondary, random, key with a *link* stored under the common key. As the key for this *index* is shared, intermediate nodes can not determine whom the packets concern. This protects the publishers, but reveals information about the user for whom they are intended, such as the number of friends and call intentions. Although a minor threat, mitigated using decoys, the scheme is inefficient and easily sabotaged, as the caller needs to process a large number of packets before finding the right one.

The most practical solution we found is to simply use shared secrets to obfuscate the storage keys. After encrypting the registration packet, the storage key is formed by appending a shared secret to the SIP AOR, and computing a hash digest (using algorithms such as MD5 [9] or SHA-1 [10]) of the result. As the hash function can not be reversed, the new key will not reveal the source or recipient of the packet.

This concludes the core of our model. Currently the static nature of the shared secrets is seen as the weakest link, although this can be mitigated using techniques such as auto-renegotiation of the shared secret, hash chains [11] or time-bound appendices. The use of these is left open, to be chosen suitably to best fit the nature of the application.

Although our model does well in protecting the privacy of users from intermediate nodes, it has a number of obvious tradeoffs. It significantly increases the amount of data stored in the overlay, as the registration packets need to be published separately for each peer. However, as the packets contain only the information necessary to contact the user, they are relatively small. We could also consider a slightly less secure linking-scheme, similar to the one described previously. The packets stored using the shared secrets would contain only an encryption key and a link and pointing to the actual registration packet, encrypted with that key.

Our model requires also that peers perform a large number of public-key cryptography operations. However, the frequency of these should be low, keeping the load well within the range of even resource-limited computing devices.

3.2 Usability Considerations

Besides the technical complications, the scheme introduces a number of usability issues that need consideration. The need for a privacy enhancing scheme might be unclear for many users accustomed to traditional communication systems operated by a trusted provider. Also, how such a scheme affects the usability of the system, how certain contacts might become unreachable as a result, might not be apparent.

When activated, we need to have both the public key and a shared secret with everyone that might wish to contact us. Otherwise we will appear unavailable, *offline*. This might seem as an extension of a presence scheme, allowing us to lay hidden from certain unwanted contacts.

The purpose is not to limit who is able to contact you, but to hide who *does*. This privacy scheme sits a level below presence, on top of which visibility can be filtered. Accordingly, although shared secrets is often associated with close relationships, the model encourages users to establish these with everyone. The aim is to affect only the visibility the sessions, not the sessions themselves.

As the scheme requires users to possess each other's public keys, as well as have established a shared secret prior to contact, initialization is a problem. Operators of high-security systems and concerned users might go through the trouble of creating such a database manually. However, most users do not appreciate the limitations and extra work, and would be satisfied with a partial open exposure, at least at times, but with the privacy enhancements used whenever possible.

The shared secrets could then be agreed on during the first contact, stored to be used for subsequent sessions.

This leads us to the question of how to present these options, *modes*, to the user. Although related to the traditional concept of presence, it has a slightly different meaning and effect, not familiar from centralized systems. In section 4.3, we present how this is solved in our implementation, discussing the pros and cons of our solution.

4 Implementation

To evaluate the feasibility of our model, and investigate the usability issues, we implemented the privacy enhancements into Helsinki Institute for Information Technology's (HIIT) Host Identity Protocol (HIP)-based P2P communication prototype.

4.1 Technology

The prototype is implemented as a light Linux-based, locally run, SIP proxy for ordinary SIP user agents (SIP UAs). It intercepts SIP signaling, converting them into the appropriate operations in the P2P model [12]. The prototype has been adopted to mobile and embedded Linux environments, and is currently used in a trial on Nokia N810 Internet tablets.

HIP [13] is a communication framework which splits the notion of identifier and locator from their union in the current Internet. HIP adds a layer between the network and transport layer, allowing applications to address end-points using cryptographically generated identifiers, instead of network addresses. HIP is used in the prototype to establish secure and robust connections between the peers, as it provides advanced features such as multihoming, mobility and NAT traversal.

HIP uniquely identifies hosts using Host Identifiers (HIs), the public part of an asymmetric key pair. As HIs are cumbersome to manage, applications use Host Identity Tags (HITs) for addressing remote hosts. These are 100 bit hashes of the HIs, which combined with the ORCHID [14] prefix are expressed as IPv6 addresses. The translation of HITs to network locations (IP addresses) is done by the underlying HIP stack, which also handles connection establishment and mobility management.

A HIP connection is established through a four-way Diffie-Hellman compliant handshake called the base exchange (BEX). During this denial of service (DoS)-protected process the hosts are authenticated and keying material is established. This keying material is used to establish a security encapsulation (using the IPSec Bound End-to-End Tunnel (BEET) mode) to protect the data traffic. As the encapsulation and routing are performed transparently by HIP, it provides advanced features (including security and IP mobility) for applications with no additional implementation effort.

4.2 Prototype Overview

As with the IETF P2PSIP protocol, the prototype adds a level of abstraction on top of the P2P overlay, accessing it through a hash table-like interface (*get*, *put* and *remove*). This allows for flexibility in its implementation, but offers no control over how the data is managed in the P2P overlay. As message routing through the overlay is not supported, it is used only to retrieve the contact information needed establish the HIP data connections between peers.

Although not the focus of the privacy model, the use of HIP could also improve privacy during the connection set-up procedure. As described in section 2, the IETF P2PSIP protocol embeds ICE address candidates in the Attach-message. Even though the content is encrypted, the NodeIds of peers are still visible, providing clues about a call. By using HIP we have a wider choice of how the connections are established. The HIP connection handshake, could be routed through the overlay, but we also have the option of using trusted relays according to the HIP NAT traversal draft [15]. In addition, the use of the BLIND extensions [16] and one-time HITs would hide the identity of the end-points.

The identity scheme of the prototype is based on trusted third-party certificates binding SIP AORs to the public keys, in line with the assumptions of section 3. Normally the SIP AORs are used as keys when storing registration packages. In addition to this *open* contact information, the privacy extensions adds the notion of *secret* storage keys, constructed using the shared secrets, under which encrypted registration packages are *hidden*.

4.3 Privacy Enhancements

The implementation of the privacy extensions included creating a local peer database and modifying how the data in the distributed storage is managed. The prototype maintains a database of the public keys and shared secrets of all the peers it has been in contact with. Although it is possible to manually configure a shared secret, a key-negotiation protocol was implemented, which automatically establishes (or reconfigures) a shared secret when connected.

This protocol is a simple three-part protocol which checks whether both peers have the same secret, and in case they differ, re-initializes one from two parts. Both peers propose one (randomly generated) part each, which are combined into the final secret. This allows for easy reconfiguration, in case either peer looses it or wants to prevent intermediate nodes from tracking these between sessions. The prototype can be configured to re-initialize these secrets each time a new session is established.

The privacy enhancements are activated using a configurable setting of the prototype, affecting how the registration packets are retrieved and published. After analyzing different use-cases, we came to the conclusion that three different modes are needed:

Open. The registration package is published using both the open and secret storage keys. Lookup is done using only the open, even when a shared secret has been established with the recipient.

Relaxed. As in the Open mode, the registration package is published using both keys. The lookup differs; if a shared secret has been established, only the secret storage key is used, otherwise the open.

Paranoid. When set to paranoid, only the secret keys are used, both when publishing and performing lookups.

The reasoning behind the Paranoid mode is understandable; the prototype uses the privacy enhancement fully, revealing as little as possible. But as discussed in section 3, for the enhancements to be usable, we also need to have an option of being openly exposed, at least temporarily. The need for two, nearly identical, modes is based on the assumption that nodes might get reset at times, loosing the database of shared secrets.

The Relaxed mode operates using a *best-effort* principle. Whenever a shared secret has been established, the privacy extensions are used without even trying the open key. By falling back to the open key, we would avoid the additional Open mode. But this could be seen as a violation of the privacy of the recipient, if in Paranoid mode. By falling back the use of open keys in the Relaxed mode, either by using subsequent or simultaneous lookups, we might reveal a relationship between the two. Although the risk of this may be low, the usability also seemed better, as the user knows beforehand whether the privacy enhancements are used. This is however something that might still change depending on the feedback we receive.

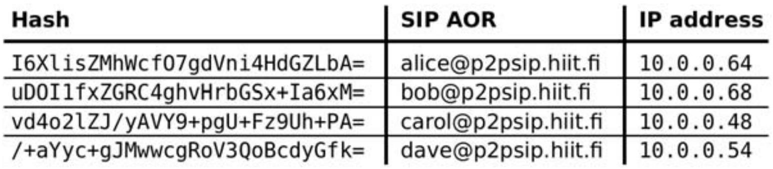
4.4 Evaluation

As a demonstration of our model, we created a small test bed of four user accounts to simulate a group of privacy conscious peers. The prototype, with the privacy enhancements, was deployed on hand-held Nokia N810 Internet tablets, with an appropriate user account configured on each. The tablets were given to a set of test users, who were asked to use them for voice- and video calls, and instant messaging.

The tablets were connected using a standard IEEE 802.11b Wireless LAN (WLAN) access point with a Dynamic Host Configuration Protocol (DHCP) server to provide IP addresses. To simulate a P2P overlay, we used LAN broadcast as the back-end for the distributed storage. To evaluate the enhancements, we recorded and analyzed the network traffic generated by the tablets, simulating the wost-case scenario where an intermediate can log all traffic. The purpose was to compare what can be deduced from the logs before and after the privacy enhancements.

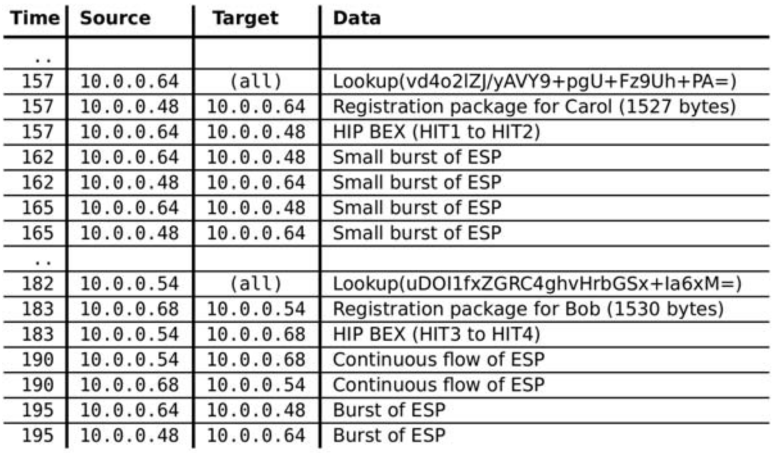
Initially the prototype was set to the Open mode. Although the keys used in the lookup are hashes of the SIP AORs, the identifiers they correspond to can be read from the clear-text response. After a short while, we had compiled a mapping of these, together with the IP address used by each peer (presented in figure 1),

A sample from the recorded traffic is presented in figure 2. From the log we see Alice establishing a connection to Carol at 157 seconds. Even though the actual data traffic is secured using HIP, the pattern of relatively small ESP burst provides clues about an instant messaging session. At 182 seconds, we see



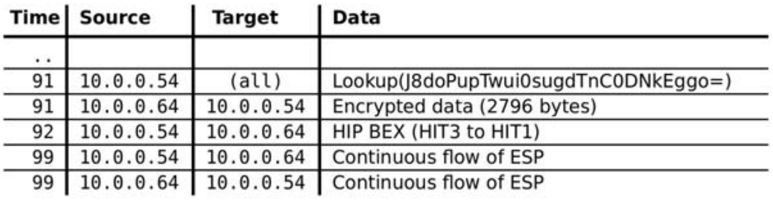
Hash	SIP AOR	IP address
I6XlisZMhWcf07gdVni4HdGZLbA=	alice@p2psip.hiit.fi	10.0.0.64
uDOI1fxZGRC4ghvHrbGSx+Ia6xM=	bob@p2psip.hiit.fi	10.0.0.68
vd4o2lZJ/yAVY9+pgU+Fz9Uh+PA=	carol@p2psip.hiit.fi	10.0.0.48
/+aYyc+gJMwwcgRoV3QoBcdyGfk=	dave@p2psip.hiit.fi	10.0.0.54

Fig. 1. The hash to SIP AOR relationships found, with responsible IP address



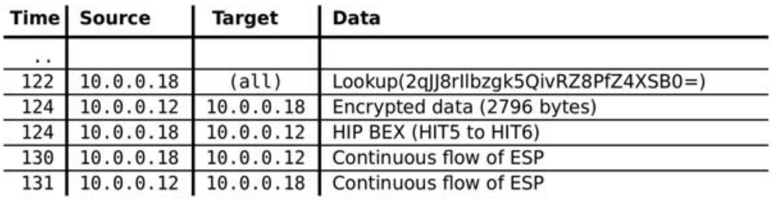
Time	Source	Target	Data
..			
157	10.0.0.64	(all)	Lookup(vd4o2lZJ/yAVY9+pgU+Fz9Uh+PA=)
157	10.0.0.48	10.0.0.64	Registration package for Carol (1527 bytes)
157	10.0.0.64	10.0.0.48	HIP BEX (HIT1 to HIT2)
162	10.0.0.64	10.0.0.48	Small burst of ESP
162	10.0.0.48	10.0.0.64	Small burst of ESP
165	10.0.0.64	10.0.0.48	Small burst of ESP
165	10.0.0.48	10.0.0.64	Small burst of ESP
..			
182	10.0.0.54	(all)	Lookup(uDOI1fxZGRC4ghvHrbGSx+Ia6xM=)
183	10.0.0.68	10.0.0.54	Registration package for Bob (1530 bytes)
183	10.0.0.54	10.0.0.68	HIP BEX (HIT3 to HIT4)
190	10.0.0.54	10.0.0.68	Continuous flow of ESP
190	10.0.0.68	10.0.0.54	Continuous flow of ESP
195	10.0.0.64	10.0.0.48	Burst of ESP
195	10.0.0.48	10.0.0.64	Burst of ESP

Fig. 2. Samples of the traffic log before the privacy enhancements



Time	Source	Target	Data
..			
91	10.0.0.54	(all)	Lookup(J8doPupTwui0sugdTnC0DNkEggo=)
91	10.0.0.64	10.0.0.54	Encrypted data (2796 bytes)
92	10.0.0.54	10.0.0.64	HIP BEX (HIT3 to HIT1)
99	10.0.0.54	10.0.0.64	Continuous flow of ESP
99	10.0.0.64	10.0.0.54	Continuous flow of ESP

Fig. 3. Samples of the traffic log with the privacy enhancements in use



Time	Source	Target	Data
..			
122	10.0.0.18	(all)	Lookup(2qJJ8rIlbzgk5QivRZ8PfZ4XSB0=)
124	10.0.0.12	10.0.0.18	Encrypted data (2796 bytes)
124	10.0.0.18	10.0.0.12	HIP BEX (HIT5 to HIT6)
130	10.0.0.18	10.0.0.12	Continuous flow of ESP
131	10.0.0.12	10.0.0.18	Continuous flow of ESP

Fig. 4. Samples of the traffic log with the IP and HIT reset, and privacy enhancements in use

how Dave contacts Bob. The continuous flow of ESP suggests a voice- or video call. At 195 seconds we see again traffic, most likely instant messages, exchanged between Alice and Carol.

After the initial session, the privacy mode was set to Paranoid, and the devices rebooted to reset any existing IPSec security associations. A short sample from the traffic is shown in figure 3. Although we can not determine what was sought at 91 seconds, it is fairly certain that a call was made between Dave and Alice. The lookup response is encrypted, but the size fits within what we would expect of an encrypted registration package, and the data traffic matches the pattern of a voice- or video call. The IP addresses (and the HITs) reveal the peers involved.

To simulate the use of additional source address hiding techniques, the HIP Host Identity database was reset on all devices, and the DHCP server configured to provide addresses from a different IP range. A sample of the traffic is shown in figure 4. We can still see that a voice- or video call is made, but the peers remain unknown.

5 Discussion

As the traffic analysis shows, the privacy enhancements does a fair job ensuring that the data managed in P2P communication systems reveal as little as possible on the application-level. However, for complete privacy, we need to consider other factors as well.

Using HIP to secure the data connections provides many benefits, but identifies the end-points to outsiders as well. Even without HIP, IP, hardware Media Access Control (MAC) addresses, or other host identification schemes might be used for the same purpose. Although the session data is encrypted, traffic analysis may reveal the type of content. However, these issues are considered out of scope for our work, as they relate to the general problem of communications privacy, and is being addressed by work such as the SlyFi [17] design. Our focus is only on the data managed by the distributed storage, and what it reveals; an issue specific to P2P systems.

The privacy model has usability issues which need to be examined further. The proposed Open, Relaxed and Paranoid modes seems reasonable from a technical point of view, but the usability of these needs to be verified.

6 Summary and Future Work

In this paper, we reviewed the differences between fully distributed and centralized communication systems, and the privacy issues in both. We presented a simple model for enhancing privacy in P2P communication systems, such as the one being developed by the IETF P2PSIP working group. The model obfuscates the data managed by the overlay-based storage by using public-key encryption and shared secrets. As it affects only the application-level data handling, it can be used in conjunction with addition lower-level techniques for greater security.

We highlighted possible usability problems, and concluded with an overview of our implementation and traffic analysis from its use, demonstrating its effect. The model does well in protecting the privacy of users on the application-level. It does increase the load on the system, but not unreasonably.

Future work will concentrate on evaluating the usability and combining the model with additional techniques to improve the overall privacy. This include examining the impact of different link-layer techniques and methods for preventing traffic pattern analysis. Also, wider trials and simulations of real user behavior, possible on a global test-bed such as PlanetLab [18], should be conducted to evaluate the feasibility of the solution.

Acknowledgments. The authors like to thank everyone associated with the design and development of the work presented in this paper. First and foremost Cheevarat Jampathom, responsible for a large share of it, but also the people involved in the TrustInet and InfraHIP projects at HIIT.

References

1. Mondal, A., Kitsuregawa, M.: Privacy, security and trust in p2p environments: A perspective. In: 17th International Conference on Database and Expert Systems Applications, pp. 682–686 (2006)
2. Good, N.S., Krekelberg, A.: Usability and privacy: a study of kazaa p2p file-sharing. In: CHI 2003: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 137–144. ACM Press, New York (2003)
3. Lu, Y., Wang, W., Bhargava, B., Xu, D.: Trust-based privacy preservation for peer-to-peer data sharing. IEEE Transactions on Systems, Man and Cybernetics 36(3), 498–502 (2006)
4. IETF P2PSIP working group, `http://www.ietf.org/html.charters/p2psip-charter.html`
5. Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H.: REsource Location and Discovery (RELOAD)(2008) (Work in progress)
6. Rosenberg, J.: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols (2007) (Work in progress)
7. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM Press, New York (2001)
8. Douceur, J.R.: The sybil attack. In: IPTPS 2001: Revised Papers from the First International Workshop on Peer-to-Peer Systems, pp. 251–260. Springer, London (2002)
9. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321 (Informational) (1992)
10. Eastlake III, D., Hansen, T.: US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634, Informational (2006)
11. Lamport, L.: Password authentication with insecure communication. Communications of the ACM 24(11), 770–772 (1981)
12. Koskela, J.: A HIP-based peer-to-peer communication system. In: ICT 2008: Proceedings of the 15th International Conference on Telecommunications, pp. 1–7 (2008)

13. Moskowitz, R., Nikander, P.: Host Identity Protocol (HIP) Architecture. RFC 4423, Informational (2006)
14. Nikander, P., Laganier, J., Dupont, F.: An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID). RFC 4843, Experimental (2007)
15. Komu, M., Henderson, T., Tschofenig, H., Melen, J., Keränen, A. : Basic HIP Extensions for Traversal of Network Address Translators (2009) (Work in progress)
16. Ylitalo, J., Nikander, P.: Blind: A complete identity protection framework for endpoints. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2004. LNCS, vol. 3957, pp. 163–176. Springer, Heidelberg (2006)
17. Greenstein, B., McCoy, D., Pang, J., Kohno, T., Seshan, S., Wetherall, D.: Improving wireless privacy with an identifier-free link layer protocol. In: MobiSys 2008: Proceeding of the 6th international conference on Mobile systems, applications, and services, pp. 40–53. ACM, New York (2008)
18. PlanetLab: An open platform for developing, deploying and accessing planetary-scale services, `http://www.planet-lab.org/`

On Modeling Viral Diffusion in Heterogeneous Wireless Networks

Hoai-Nam Nguyen* and Yoichi Shinoda

Japan Advanced Institute of Science and Technology
#1-518, 1-8 Asahidai, Nomi, Ishikawa, Japan 923-1211
{namnh,shinoda}@jaist.ac.jp

Abstract. Smart phones and computers now are able to co-work in a wireless environment where malware can propagate. Although many investigations have modeled the spread of malware, little has been done to take into account different characteristics of items to see how they affect disease diffusion in an ad hoc network. We have therefore developed a novel framework, consisting of two models, which consider diversity of objects as well as interactions between their different classes. Our framework is able to produce a huge result space thus makes it appropriate to describe many viral proliferating scenarios. Additionally, we have developed a formula to calculate the possible average number of newly infected devices in the considered system. An important contribution of our work is the comprehension of item diversity, which states that a mixture of device types causes a bigger malware spread as the number of device types in the network increases.

1 Introduction

Malware such as viruses, worms, or malicious codes/programs has long posed a significant threat to computers, regardless of fixed or mobile ones, but so far mobile phones have been relatively safe. Recently, however, attacks by various types of viruses, trojans, and spyware on mobile phones, especially smart phones, have been increasing. Reports about worms such as **cabir** [2], [5] and trojans such as **skulls** and **mquito** [5] raise serious questions about mobile phones' vulnerabilities. Such malicious codes[1] are able to infect many users under appropriate conditions. For example, a sports event in Helsinki in August 2005 and a public concert in Germany, Live8, were marked by outbreaks of viruses [10]. Another peril when a phone is under control of a hacker, which can be accomplished using a virus, is described in [16], [17]. They state that it is possible for a malicious user to break down a phone network using SMS messages.

Smart phones, which have been undergoing an explosive growth in sheer numbers, along with the commensurate improvements in computing ability and generality, have become an attractive target for malware. Improvements in hardware and software for phones, especially their communications capabilities, create a myriad of avenues for

* Corresponding author.

[1] We will also call virus, worm, and trojan malicious code, malicious program, malware, or malcode if not specifically mentioned.

hackers to exploit and attack. There is malware capable of dwelling in both computers and phones so it is certainly possible for malcode to hit phones from computers and vice versa. In general, then, malicious threats can infect computers or phones in one of the followings ways [3]:

(1) Through communication interfaces such as Bluetooth, Wifi, or infrared. A malware can directly contaminate other devices whenever they are reachable and connectable and they have exploitable weaknesses. We count computers that have wireless interfaces as susceptible objects of contagion as well. Note that, for this propagation relation, we have to consider the bidirectional effect, since a malicious program can move not only from computer to computer but also from computer to phone and vice versa.
(2) Through user download programs infected with the malware.
(3) By using the Multimedia Messaging Service (MMS) to diffuse to other phones.
(4) By a memory card which already contains a malicious program. When the user uses that card in a phone or computer, the compromised program transmits itself to the device.

Of the above ways, only (3) is restricted to phone-phone relations, whereas the others are risky for both computers and phones.

In addition to the improved device technologies, the mobility of users is another factor that makes malware spread wider and faster. Indeed, when a user with an infected device is on the move, he/she is likely to make more contacts, thus giving malicious code an opportunity to transfer to more hosts.

In this paper, we develop a novel analytical framework, by which we are able to explore how viral transmissions spread through a network containing devices that are heterogeneous in terms of various characteristics (such as device types). Here in our framework, each network item experiences states from susceptible to infected. These states with variability and changes are carefully described by mathematical parameters. One of the most advantageous points of our models is their huge result space. Indeed, if we use different combinations of parameters, we consequentially get various outputs that correspond to many spreading scenarios of malware. Our framework also takes into account all the contamination methods mentioned above.

We then use that framework to derive the necessary conditions for the existence and persistence of malware. With the framework, it is able for us to predict the possible average number of newly infected individuals and that may assist network administrators to have a right strategy to combat or mitigate the malware. Investigations of malware free equilibrium and its locally asymptotic stability are additional contributions of our work. Moreover, we check whether diversity of participant characteristics in a network has any influence on virus/worm propagation. It is quite intuitive to see that viral dissemination is wider if items in a network possess more characteristics, but we have gone further to theoretically prove this statement.

The rest of this paper is organized as follows: in Section 2, we review and discuss the related works. This section also differentiates our work from other studies reported in the literature. Next, Section 3 describes the deterministic analytical models in depth. We present a detailed discussion of the proposed models in Section 4. Numerical results of conducted simulations are given in Section 5. Finally, Section 6 contains concluding remarks and outlines our future work.

2 Related Works and Motivation

Traditional networks with fixed nodes connected through wired media have suffered many serious virus/worm outbreaks such as those created by the Melissa [1] or Codered [7], [20] worms. With improvements in advanced wireless technology, networks with laptops having wireless interfaces have grown rapidly in both quantity and complexity. A group of nodes that communicate via wireless interfaces, forming a network without any pre-defined topological structure, is called a wireless ad hoc network. In this type of network, a node can move freely and contact other nodes through wireless interfaces. Modeling the spread of malware on this type of network has received considerable attention from researchers, [6], [9] to name a few. Nevertheless, these works have not taken smart phones into account, a new source of inspiration for attackers.

Ramachandran and Sikdar [13], [14] have studied the spread of malware in a phone network, assuming that nodes move from one patch to the others at a specific rate. They incorporate movement, or in their words, ”the heterogeneity of locality,” into their model to make it more precise. However, they do not take into consideration the inner characteristics of each class of objects in the network. They indirectly suppose the same condition for all devices by delivering the same values for parameters. As a result, the model is unable to distinguish individual classes if diverse entities exist concurrently. In addition, these studies do not contain an analysis of the relationship between the number of patches and the spread of the disease.

One of our published papers, [11], proposes a 4-compartment model to describe how viral forces propagate in a wireless network. We used the same reasoning and observation but adding one more state that a susceptible item may experience while participating in a network. The added latent state is understandable as the incubation period of the virus. That paper, however, does not provide an in-depth investigation on the latent state such as calculating the average time a network item has to be in if it is exposed to the spreading malware.

Proposing a model that will depict all characteristics of a set of connectable and reachable nodes, computers and phones, is a difficult task since we have to include as many parameters as possible. In our research, one major input of interest is the fundamental difference between classes of objects in the network[2]. We classify and arrange items in a network into different groups based on their characteristics. It is understandable as item diversity and we aim to fill the gaps of not taking care of device diversity left by previous research. With that goal, our models take the following aspects into consideration:

– **heterogeneity of entity**, i.e., the existence of different types of devices in the network

[2] For ease of reading, we now refer to a network either as a collection of computers and phones, computers only, or phones only, that all reside in a considered territory.

- interactions between different classes of entities. As a matter of fact, there are many types of operating system (OS) for smart phones and each of which has its own particular vulnerabilities for attackers to exploit. Consequently, we count phones with the same OS as items in the same class while those with different ones are put into different classes. Note that computers are recognized as a separate set.
- diversity of entities. We examine whether diversity has an impact on the persistence and stability of a disease as well as its spread dynamics.

3 Modeling Disease with Heterogeneity of Nodes

We built up our model using a typical compartmental epidemic model for two cases, each of which has three entity classes. Under different assumptions, the last class in each model differs while the rest remain the same. Here is the description of the two cases: at an arbitrary point in time, a device is in one of the following three classes: susceptible (S), infected (I), and moved back to susceptible state (S) for the first case or recovered and immune to re-infection (R) for the second case. Every object in the network is initially susceptible to disease if it is not already infected. The object stays in the susceptible phase until it comes into contact with an infectious one or with the malware. As soon as the malware propagates to a susceptible item, the item moves to the infected state. For the first case, when a device is cured of disease, it recovers and therefore is subject to re-infection if no immunity has been introduced. This case is the SIS model. In the second case, after the end of contamination, a device produces resistance to the disease and is removed from the infected condition and does not progress back to susceptible but recovered and immune class. The SIR model represents this case.

3.1 Assumptions

In this section, we state our assumptions. We assumed that multiple items can be infected but one species is the primary infectious vector. Since computers suffer by far the most attacks and they outnumber other devices in terms of the diversity of malware, we made computers the main attack force of offenders while the others are spillover species. At the moment, computer viruses do not usually try to propagate to mobile phones. However, as malware develops, it is clear that this could be a forthcoming problem, possibly in the near future [3]. Because of the higher possibility that malcodes transfer from computers to smart phones compared to that from the reverse direction, we chose computers to be the main disease reservoir. Another assumption we made is that computers in a network are of the same kind. Worms often exploit holes in software and there are many applications that run on both computers and phones under different operating systems. Accordingly, malware is supposedly able to diffuse to nodes using distinct operating systems.

When several phone users coincidentally gather in an area and make calls or send messages, it is likely that those overcrowded occurrences will overwhelm the capacity of the network. It means that as more phones come into the congestion, the less service they receive from the operator, and thus infection vectors for phones such as MMS or dialing are closed off. The same thing may happen to computers as well, but in a dissimilar scenario. For example, in an ad hoc network where data moves from computer to computer in a hop-by-hop manner, there is a possibility that one computer would serve as the main transporter between two components of the network. The more data sent through this bottleneck, the greater the risk that this computer will become overloaded and break down. At a certain level, the collapse of this computer will result in a disconnected component from the network. All items in that detached component are considered dead. The above scenarios lead us to the following definition: The number of devices that malware can not use for propagation due to the congestion is called *density-dependent death rate*.

3.2 Elaborate Models

In this section, we first derive the SIS epidemic model and then formulate the SIR model for n different types of hosts. As mentioned, one species is considered the main reservoir while the rest, $n-1$ species, act as spillover ones.

Let S_1 and I_1 denote the susceptible and infected reservoir population and S_k and I_k the susceptible and infected spillover populations, $k=2,\ldots,n$. Therefore, the total population size for each type k is $N_k = S_k + I_k$, $k=1,\ldots,n$. We then find an equation to describe the changes in the susceptible set of each population. We denote by b_k the rate at which each species has more entities, $k=1,\ldots,n$. In other words, b_k illustrates the birth rate of the kth type. The total number of new-born units in the kth group is then calculated by $N_k b_k$.

Next we denote by $d_k(N_k)$ the density-dependent death rate of the kth type. Note that this parameter depends on the total population size of the kth group. In the following formulae, we shorten it by using d_k. Consequently, the quantity of out-of-order disease-unrelated units from S_k is $S_k d_k$.

To demonstrate the disease-related interactions between groups through communicating interfaces such as Wifi, Bluetooth, or infrared, we define a factor $\alpha_{ki}(N_i)$, dependent on N_i, for each pair of device types k and i, $k=1,\ldots,n$ and $i=1,\ldots,n$. This rate is understood as an infected item in group i tries to infect a susceptible individual in group k. The rate magnitude of each type i is proportional to the ratio of infective pieces over the total population size, which is given by $\frac{I_i}{N_i}$. Scanning through all populations, the S_k group loses total $S_k \sum\limits_{i=1}^{n} \alpha_{ki} \frac{I_i}{N_i}$.

When a user downloads a program containing malware and incautiously executes that program, the device in use (computer or phone) directly moves to the infective target compartment. Let β_k be the rate at which items in S_k download some program from the Internet and let γ be the probability that a program contains malware. For ease of analysis, we assume that when a device is infected by a specific malware, it will not be compromised by other malcodes. Accordingly, malware only affects susceptible items but not already infected ones. This argument gives us the number of entities that move from S_k to I_k is $\beta_k \gamma S_k$.

The third mechanism through which malware can be dispersed is use of the MMS service. A compromised phone may randomly choose a number from the contact list or generate a random number to dial then send the malware enclosed in an MMS message. The rate of infected devices in the kth type that try to dial other numbers is referred to as η_k. Nonetheless, some of the dialed numbers are out-dated (if chosen from contact list) or do not exist (if randomly generated) so those attempts may fail sometimes. For that reason, we use ϑ as the probability that the malware succeeds in sending messages to other numbers. We also note that when the target phone is off, it belongs to the exposed class since there is a poisoned piece awaiting it in the queue and when it is on, the phone directly moves to the infected class. However, when we count total loss of the susceptible population, this method of dissemination shifts items from susceptible to exposed and infected classes a change of $-\eta_k \vartheta S_k \frac{I_k}{N_k}$.

The last carrier for malware is memory cards. Analogously, we call μ_k the rate at which users in susceptible group S_k use memory cards and τ the probability that a card contains at least one malware. Hence, this carrier subtracts from S_k an amount of $\mu_k \tau S_k$.

Finally, the recovery rate, the rate at which devices are patched, of type k is denoted by φ_k so the total number of recovered pieces is $\varphi_k I_k$.

Taking all into account, changes in S_k, $k = 1, \ldots, n$, can be formulated as follows:

$$\frac{dS_k}{dt} = N_k b_k - S_k d_k + \varphi_k I_k - S_k \left(\sum_{i=1}^{n} \alpha_{ki} \frac{I_i}{N_i} + \beta_k \gamma + \eta_k \vartheta \frac{I_k}{N_k} + \mu_k \tau \right). \quad (1)$$

We use λ_k to demonstrate the disease-related death rate at which infected entities are inoperative by the malware. With the same reasoning the following equation tells us about variations in each I_k, $k = 1, \ldots, n$:

$$\frac{dI_k}{dt} = -I_k d_k - (\varphi_k + \lambda_k) I_k + S_k \left[\sum_{i=1}^{n} \alpha_{ki} \frac{I_i}{N_i} + \beta_k \gamma + \eta_k \vartheta \frac{I_k}{N_k} + \mu_k \tau \right]. \quad (2)$$

Equations (1) and (2) form the SIS epidemic model.

To obtain the SIR model, we add a compartment of entities that produce immunity after disease recovery. For each type k, $k = 1, \ldots, n$, a class of recovered and immune items, R_k, is added to the existing model (1) and (2) . Similarly, we build up the SIR model as follows:

$$\frac{dS_k}{dt} = N_k b_k - S_k d_k - S_k \left(\sum_{i=1}^{n} \alpha_{ki} \frac{I_i}{N_i} + \beta_k \gamma + \eta_k \vartheta \frac{I_k}{N_k} + \mu_k \tau \right), \quad (3)$$

$$\frac{dI_k}{dt} = -I_k d_k - (\varphi_k + \lambda_k) I_k + S_k \left[\sum_{i=1}^{n} \alpha_{ki} \frac{I_i}{N_i} + \beta_k \gamma + \eta_k \vartheta \frac{I_k}{N_k} + \mu_k \tau \right], \quad (4)$$

$$\frac{dR_k}{dt} = -R_k d_k + \varphi_k I_k. \quad (5)$$

All parameters are likewise understood as in the SIS model, except for the φ_k one, which accounts for the rate at which infected entities recover and move to immune compartment R_k. The total population is the sum of the three compartments, that is: $N_k = S_k + I_k + R_k$, $k = 1, \ldots, n$.

Note that all parameters are assumed to be greater than or equal to zero.

4 Discussion on the Models

4.1 Assumption and Parameter Discussion

In this part, we further discuss about the assumptions, parameters, and the models. To distinguish the dynamics of the main reservoir from other species, we assume a higher intrinsic transmission and scanning rate from the reservoir than from the spillover species or between groups of spillover species. Mathematically that is,

$$\alpha_{11}(N_1) \geq \alpha_{k1}(N_1) \geq \begin{cases} \alpha_{ki}(N_i), & k, i = 2, \ldots, n \\ \alpha_{1i}(N_i), & i = 2, \ldots, n. \end{cases} \tag{6}$$

When no infection occurs, we have $\lim_{t \to \infty} N_k(t) = H_k$, where H_k is the carrying capacity for species k. Carrying capacity is the maximum number of individuals that a system can support without degradation. This matches our assumption of the natural density-dependent death rate.

Next we derive the disease-free equilibrium (DFE) for each model. The DFE for (1) and (2) is the unique solution satisfying $S_k^* = H_k$ and $I_k^* = 0$, $k = 1, \ldots, n$. Thus the equilibrium point, Q, is:

$$Q_0 = (S_1^*, I_1^*, \ldots, S_n^*, I_n^*) = (H_1, 0, \ldots, H_n, 0). \tag{7}$$

On the contrary, a nonnegative endemic equilibrium point Q_1 exists with $I_k^* > 0$ which requires that $I_i^* > 0$ when $\alpha_{ik}(N_k) > 0$. That means if there is transmission between two species groups, the persistence of the disease in one can result in persistence of the disease in the other. The DFE for model (3), (4), and (5) is $S_k^* = H_k$ and $I_k^* = R_k^* = 0$, $k = 1, \ldots, n$.

4.2 Analysis of Disease Extent

The malware may have an opportunity to spread throughout the network or it may die out at a certain time. In this section, we study the stability of the disease and derive the necessary conditions for its stable state. We compute the basic reproduction number $\mathcal{R}_0$ to examine the mean number of possible secondary infected cases introduced into a susceptible population by a single infected case [8]. This number describes the possible average number of newly infected devices and it will be proven that this number is identical for the proposed SIS and SIR models.[3]

Theorem 1. *Let $\rho(M)$ denote the spectral radius of matrix M. The possible average number of newly infected items, $\mathcal{R}_0$, for models (1), (2) and (3), (4), (5) is given by the spectral radius of the matrix $M_n = (\mathcal{R}_{ij})_{n \times n}, i, j = 1, \ldots, n$,*

$$\mathcal{R}_0 = \rho(M_n),$$

[3] "Basic reproduction number" and "average number of newly infected units/item/entities/individuals" will be used interchangeably.

where

$$\mathcal{R}_{ij} = \begin{cases} \dfrac{\alpha_{ii} + \eta_i \vartheta}{b_i + \varphi_i + \lambda_i} & \textit{if } i = j, \\ \dfrac{H_i \alpha_{ij}}{H_j(b_j + \varphi_j + \lambda_j)} & \textit{otherwise,} \end{cases} \tag{8}$$

is the ijth entry in the matrix M_n, $i, j = 1, \ldots, n$.

The basic reproduction number functions as a threshold parameter in understanding the state of disease. It is proven in [18] that if $\mathcal{R}_0 < 1$, then the DFE is locally asymptotically stable, meaning that the disease does not spread through the whole network, but if $\mathcal{R}_0 > 1$, the DFE is unstable and the disease may break out.

The following theorem gives us an important view by which we have a more profound understanding of malware spread in a heterogeneous wireless network.

Theorem 2. *Assume that the average number of newly infected units $\mathcal{R}_0$ for a n-type system of models* (1)*,* (2) *and* (3)*,* (4)*,* (5) *is given by the spectral radius of the $n \times n$ matrix M_n defined in Theorem 1. If one more spillover species comes into the existing network and all the parameters in the equations remain unchanged, the new SIS and SIR models with $(n+1)$ types of items have the basic reproduction number, i.e., the spectral radius of the $(n+1) \times (n+1)$ matrix M_{n+1}, $\rho(M_{n+1})$, that satisfies*

$$\rho(M_{n+1}) \geq \rho(M_n) \geq \mathcal{R}_{11}. \tag{9}$$

An expansion for the Theorem 2 is as follows:

Claim. Additionally, if the transmission rate for the kth species satisfies

$$\alpha_{k1}(H_1)\alpha_{1k}(H_k) \neq 0, \qquad k \neq 1, \tag{10}$$

then

$$\rho(M_n) > \mathcal{R}_{11}, \qquad \forall n > 1. \tag{11}$$

Proofs for these theorems and claim are provided in the appendices.

Theorem 2 gives us an important result: If a new population is introduced into an existing system and this new population has disease-related transmissions back and forth with populations that are already in the system, the possible average number of infected devices tends to increase. That means, the more types of individuals appear in the network, the higher the probability that an epidemic infection spreads wider. Although it is straightforward to see this, we have set a theoretically sound base by the above theorem. We will show more evidence in the simulation section where the numerical outcomes totally agree with what stated here.

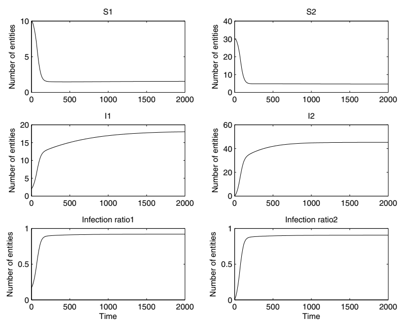
5 Simulation Results

In this section, we evaluate several examples to study the dynamics of the SIS and SIR models. Examples had unlike combinations of parameters to enhance the theoretical findings. According to the first setting, we assumed that a malcode is newly emerged and it is able to massively spread. The same problem of serious spread has been found in traditional fixed networks [15], [19] and we raise awareness about it in the wireless environment and tested by our models. To reflect the massive diffusion, high contact rates and low recovery rates were employed. In the first simulation, we tested the variations in a network with two device types: computers and phones with a specific operating system, say OS1. Let $b_1 = 0.3$ and $b_2 = 0.5$ for the birth rate. We used a simple form for the density-dependent death rates, $d_1(N_1) = 0.1 + 0.01N_1$ and $d_2(N_2) = 0.1 + 0.008N_2$. Attempts to transfer malware by WLAN or Bluetooth were modeled by $\alpha_{ij}, i, j = 1, 2$ as follows: $\alpha_{11} = 0.15N_1,\ \alpha_{12} = 0.05N_2,\ \alpha_{21} = 0.08N_1,$ and $\alpha_{22} = 0.05N_2$. The download rates for two species are $\beta_1 = 5$ and $\beta_2 = 2$; the probability that a program is malware is $\gamma = 0.005$. Since MMS is only valid for phones, we let $\eta_1 = 0$ for computers and $\eta_2 = 5$ for phones; and the probability a phone is successful in sending the compromised message is $\vartheta = 0.1$. We then describe the last infection mechanism through memory cards. Assuming that computer users and phone users use memory cards at the same rate, let $\mu_1 = \mu_2 = 3$. The probability a memory card contains malware is $\tau = 0.1$.

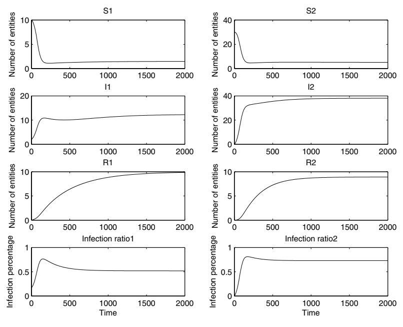
Antivirus softwares for computers are updated faster than for phones and computer users tend to scan their computers more frequently, so the recovery rates for computers and phones were set at $\varphi_1 = 0.2$ and $\varphi_2 = 0.1$, respectively. Nowadays, less malware, upon usurpation on victims, actually causes the sufferers to break down . Thus, we set the disease-related death rates $\lambda_1 = \lambda_2 = 0.1$.

The possible average number of infected items $\mathcal{R}_0$ for the SIS model therefore had a value of 9.7. As we can see in Fig. 1(a), the infected populations, I_1 and I_2, go up in the initial period while the susceptible populations sharply decrease because of the massive infection. They all stabilize at certain numbers during the simulation. This is in agreement with the value of $\mathcal{R}_0$, which indicates that the disease had obviously prevailed.

The second simulation gives us a view of the SIR model with two types of devices, the same as in the first simulation. All parameters in the first simulation were kept unchanged. Outcomes in Fig. 1(b) showed that all compartments soon reach their stable



(a)



(b)

Fig. 1. Numerical results for massive spreads in 2 species of the: (a) SIS model with $S_1^0 = 10,\ S_2^0 = 30,\ I_1^0 = 2,$ and $I_2^0 = 0$; (b) SIR model with $S_1^0 = 10,\ S_2^0 = 30,\ I_1^0 = 2, I_2^0 = 0, R_1^0 = 0,$ and $R_2^0 = 0$

states. Similar to the SIS model, the infectious population in both computers and phones existed and persisted.

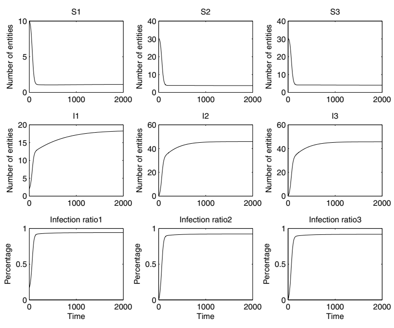
In the third and fourth numerical simulations, we tested the SIS and SIR having one more newly added type of phone with operating system OS2. All parameters in the system were kept unchanged and we used the following extra parameters in need caused by the addition: The density-dependent death rate for the third entity is $d_3 = 0.1 + 0.008N_3$. Attempt rates to transfer malware through wireless interfaces, given in matrix form, was:

$$\alpha = \begin{pmatrix} 0.15N_1 & 0.07N_2 & 0.05N_3 \\ 0.08N_1 & 0.08N_2 & 0.03N_3 \\ 0.08N_1 & 0.03N_2 & 0.07N_3 \end{pmatrix}.$$

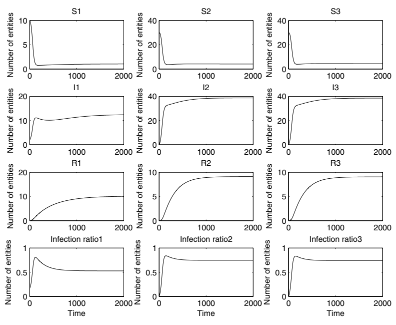
Items of this type download with the rate $\beta_3 = 2$. We let the rate at which compromised phones of the new type send MMS messages be 5, i.e., $\eta_3 = 5$. Users using this type of phones were assumed to use memory cards at the rate $\mu_3 = 3$. This kind of phone had the recovery rate of 0.1, denote by $\varphi_3 = 0.1$. The disease-related death was set $\lambda_3 = 0.1$.

For the two latter simulations, the average number of newly infected items was 12, bigger than that for the system with two entity types. We can see the same variation patterns of populations from Fig. 2(a) and 2(b) as what had been observed in the former simulations. However, if we compare SIS with SIS (Fig. 1(a) vs. Fig. 2(a)) and SIR with SIR (Fig. 1(b) vs. Fig. 2(b)) results obtained from simulations for 2-type and 3-type networks, respectively, there are disparities. The portion of infected entities during time in both species, computers and phones with OS1, from Fig. 1(a) and Fig. 1(b) are smaller than that from Fig. 2(a) and Fig. 2(b), respectively paired as mentioned above.

We also conducted other numerical simulations to assess the influence of device diversity on viral dissemination. As the theoretical result has shown, the new system forming from the existing one with at least one more new device kind has rising average number of newly infected entities, no matter how big the recovery rate of the new network items. For both models having 3 entity types, we changed the recovery rate of

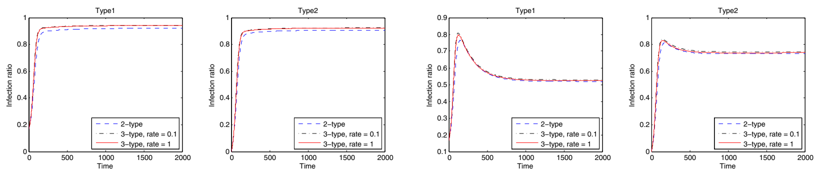


(a)



(b)

Fig. 2. Numerical results for massive spreads in 3 species of the: (a) SIS model with $S_1^0 = 10,\ S_2^0 = 30,\ S_3^0 = 30,\ I_1^0 = 2, I_2^0 = 0$, and $I_3^0 = 0$; (b) SIR model with $S_1^0 = 10,\ S_2^0 = 30,\ S_3^0 = 30,\ I_1^0 = 2, I_2^0 = 0, I_3^0 = 0, R_1^0 = 0, R_2^0 = 0$, and $R_3^0 = 0$



(a) (b)

Fig. 3. Analysis of different recovery rates of the third species for the (a) SIS and (b) SIR model with massive spread

the third population from 0.1 to 1. The portion of infected items over the total number of items for computers and phones with OS1, as theoretically judged, are bigger than that for the network with 2 types of items. We report this difference in Figs. 3(a) and 3(b). The bottommost line shows that the portion of infected entities for two first device types in a 2-type network is smaller than that in a 3-type system. These numerical results strongly agree with the new value of the basic reproduction number that when a new entity kind is added into an existing system and there are transmissions between this new type and existing ones, the disease is likely to span to a wider extent.

In the second scenario, the malware had lower speed of spreading and users were more experienced in mitigating the viral contagion. Therefore, the wireless contact rates had lower values while the recovery rates had higher ones. All parameters are given in matrix or vector forms in Table 1. The outcomes are shown in Figs. 4(a) and 4(b). The resulted average number of newly infected items is calculated, $\mathcal{R}_0 = 0.32$. A small value of $\mathcal{R}_0$ infers that the disease has no opportunity to prevail.

Table 1. Parameters for the second scenario with a 2-type network

Parameters	**Values**
b	$[0.3, 0.5]$
d	$[0.1 + 0.01N_1, 0.1 + 0.008N_2]$
α	$\begin{pmatrix} 0.015N_1 & 0.005N_2 \\ 0.008N_1 & 0.005N_2 \end{pmatrix}$
β	$[5, 2]$
γ	0.005
η	$[0, 2]$
ϑ	0.1
μ	$[3, 3]$
τ	0.1
φ	$[1.5, 1]$
λ	$[0.01, 0.01]$

We analyzed the same problems when one more type of phone appears with the network and when this new type has variable recovery rates. Additional parameters are:

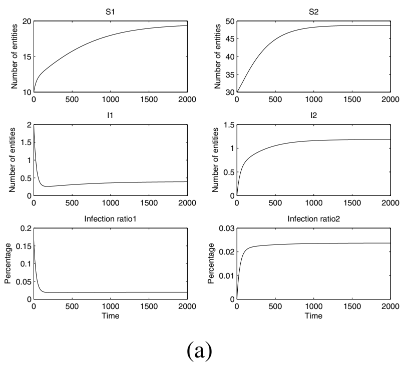
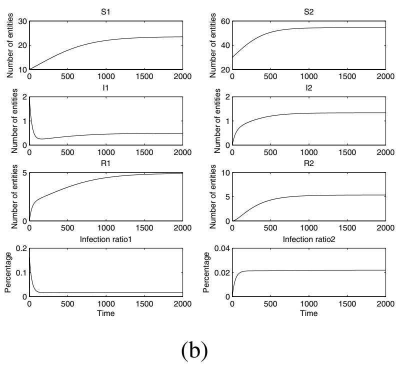



Fig. 4. Numerical results for mitigated spreads in 2 species of the: (a) SIS model with $S_1^0 = 10,\ S_2^0 = 30,\ I_1^0 = 2$, and $I_2^0 = 0$; (b) SIR model with $S_1^0 = 10,\ S_2^0 = 30,\ I_1^0 = 2, I_2^0 = 0, R_1^0 = 0$, and $R_2^0 = 0$

$d_3 = 0.01 + 0.008N_3, \beta_3 = 2, \eta_3 = 5, \mu_3 = 3, \varphi_3 = 1$, and $\lambda_3 = 0.1$. Wireless contact rates in matrix form is

$$\alpha = \begin{pmatrix} 0.015N_1 & 0.005N_2 & 0.005N_3 \\ 0.008N_1 & 0.005N_2 & 0.003N_3 \\ 0.008N_1 & 0.003N_2 & 0.004N_3 \end{pmatrix}.$$

Outcomes for this parameter combination are illustrated in Figs. 5(a) and 5(b). In this case, the reproduction number is $\mathcal{R}_0 = 0.33$, just a little bit bigger than that of a 2-type system. An explanation is that all propagating mechanisms had small rates and probabilities so the number of infected individuals did not grow much.

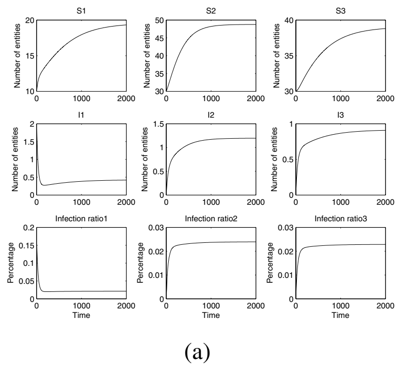
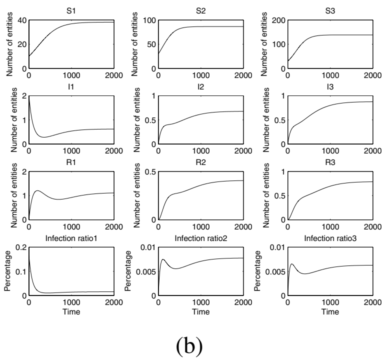



Fig. 5. Numerical results for mitigated spreads in 3 species of the: (a) SIS model with $S_1^0 = 10,\ S_2^0 = 30,\ S_3^0 = 30,\ I_1^0 = 2, I_2^0 = 0$, and $I_3^0 = 0$; (b) SIR model with $S_1^0 = 10,\ S_2^0 = 30,\ S_3^0 = 30,\ I_1^0 = 2, I_2^0 = 0, I_3^0 = 0, R_1^0 = 0, R_2^0 = 0$, and $R_3^0 = 0$

We changed the recovery rate of the third item kind to $\varphi_3 = 5$ to see the influence of item heterogeneity. In these conductions, the total portion of infected items for computers and phones with OS1 were higher than that of a 2-type network as easily seen in Figs. 6(a) and 6(b).

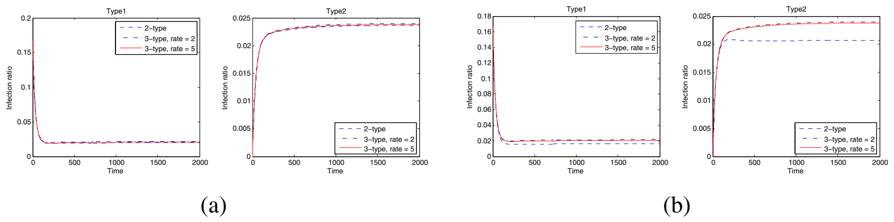


Fig. 6. Analysis of different recovery rates of the third species for the (a) SIS model and (b) SIR model with mitigated spread

6 Conclusions and Future Works

In this paper, we have examined the situation that wireless devices having different characteristics exist in a network with threats of viral infection. Various sorts of items, e.g. computers and phones, are distinguished and taken into account to precisely formulate two novel analytical models which we propose as an aid to understanding the dynamics of the spread malware through an ad hoc network. Additionally, we have derived a method to calculate the basic reproduction number, understandable as the possible average number of newly infected items, and used this number to acquire a grasp of disease diffusion as well as its stability. We also have theoretically contributed to comprehending the influence of entity diversity on malware propagation: The more distinct populations, with vulnerabilities to malware infection, introduced into an existing network, the bigger the viral dissemination is. Moreover, a huge result space is also producible by our 3-compartment framework thus makes it appropriate to describe many viral proliferating scenarios. The correctness of our models is affirmed by several numerical results under various situations.

Our work has some restrictions as follows: The simulations were conducted based on numerical results only and we did not use real network traces, and our models can be more general if we consider more states that each network item may undergo while participating in the network.

We intend to further scrutinize the basic reproduction number so that we can usefully utilize it for an investigation of the spread of malware and add mobility into our model to make it better reflect the real world. Another extension of our paper can be made on considering multiple susceptible and infected reservoirs, this could happen in reality.

Acknowledgements

We gratefully appreciate the support from the 21st century COE program "Verifiable and Evolvable e-Society," funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology. We thank Dr. Yasuhiro Ohara (JAIST) for his insightful comments to improve the quality of this paper.

References

1. Chen, T.M., Robert, J.M.: Worm epidemics in high-speed networks. Computer 37(6), 48–53 (2004)
2. Coursen, S.: The future of mobile malware. Network Security 2007(8), 7–11 (2007)
3. Hypponen, M.: Mobile malware. Invited talk, 16th USENIX Security symposium, http://www.usenix.org/events/sec07/tech/hypponen.pdf (accessed, October 2008)
4. Lancaster, P., Tismenetsky, M.: The theory of matrices: with applications, 2nd edn. Academic Press, London (1985)
5. Leavitt, N.: Mobile phones: the next frontiers for hackers? Computer 38(4), 20–23 (2005)
6. Mickens, J.W., Noble, B.D.: Modeling epidemic spreading in mobile environments. In: ACM Workshop on Wireless Security (2005)
7. Moore, D., Shannon, C., Claffy, K.: Code-red: A case study on the spread and victims of an Internet worm. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement (2002)
8. Murray, J.D.: Mathematical biology: I. An introduction. Springer, Heidelberg (2002)
9. Nekovee, M.: Worm epidemics in wireless ad hoc networks. New Journal of Physics 9 (2007)
10. BBC news service. Mobiles get anti-virus protection, http://news.bbc.co.uk/2/hi/technology/4207476.stm (accessed, Octorber 2008)
11. Nguyen, H.-N., Shinoda, Y.: A novel analytical framework to model malware diffusion in heterogeneous wireless networks. In: Proceedings of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (2009)
12. Ortega, J.M.: Matrix theory: A second course. Plenum Press, New York (1987)
13. Ramachandran, K., Sikdar, B.: Modeling malware propagation in networks of smart cell phones with spatial dynamics. In: Proceedings of IEEE INFOCOM 2007 (2007)
14. Ramachandran, K., Sikdar, B.: On the stability of the malware free equilibrium in cell phones networks with spatial dynamics. In: IEEE International Conference on Communications, ICC 2007 (2007)
15. Su, J., Chan, K.K., Miklas, A.G., Po, K., Akhavan, A., Saroiu, S., Lara, E.D., Goel, A.: A preliminary investigation of worm infections in a bluetooth environment. In: Proceedings of the ACM Workshop on Rapid Malcode, WORM (2006)
16. Traynor, P., Enck, W., McDaniel, P., LaPorta, T.: Mitigating attacks on open functionality in SMS-capable cellular networks. In: Proceedings of MobiCom 2006 (2006)
17. Traynor, P., McDaniel, P., LaPorta, T.: On attack causality in Internet-connected cellular networks. In: Proceddings of 16th USENIX Security Symposium (2007)
18. van den Driessche, P., Watmough, J.: Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. Mathematical Biosciences 180, 29–48 (2002)
19. Wierman, J.C., Marchette, D.J.: Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction. Computational Statistics and Data Analysis 45(1), 3–23 (2004)
20. Zou, C.C., Gong, W., Towsley, D.: Code red worm propagation modeling and analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (2002)

A Proof for Theorem 1

Result from [18] can be directly applied to get a proof for this Theorem.

B Proof for Theorem 2

We augment the matrix M_n with one row and one column of zeros, the resulted matrix M_n^0 has the following form:

$$M_n^0 = \begin{pmatrix} M_n & \mathbf{0}^T \\ \mathbf{0} & 0 \end{pmatrix},$$

where $\mathbf{0}$ is the $1 \times n$ vector. The next generation matrix for the new system of $(n+1)$ species is built as follows:

$$M_{n+1} = \begin{pmatrix} M_n & M_{*,n+1} \\ M_{n+1,*} & \mathcal{R}_{n+1,n+1} \end{pmatrix},$$

where $M_{*,n+1}$ and $M_{n+1,*}$ are the column and row vector with corresponding elements computable from Theorem 1, respectively. The spectral radius of M_{n+1}, $\rho(M_{n+1})$, is the basic reproduction number of the new system consisting of $(n+1)$ types of entities. With the assumptions made for the SIS and SIR models, it follows that the column and row vectors $M_{*,n+1}$ and $M_{n+1,*}$ of M_{n+1} have nonnegative entries. Consequently, from the theory of nonnegative matrices [4], [12], we have

$$\rho(M_n) = \rho(M_n^0) \leq \rho(M_{n+1}).$$

When $n = 1$, $\rho(M_1) = \mathcal{R}_{11}$. Hence, the inequality (9) holds.

C Proof for Claim

In order to prove the strict inequality (11), we see that, according to the assumption (10), $\mathcal{R}_{1k} > 0$ and $\mathcal{R}_{k1} > 0$ since they can not be equal to zero. Let $k = 2$ and consider the basic reproduction number of a two-species system, represented by the spectral radius of M_2, $\rho(M_2)$, where

$$M_2 = \begin{pmatrix} \mathcal{R}_{11} & \mathcal{R}_{12} \\ \mathcal{R}_{21} & \mathcal{R}_{22} \end{pmatrix}.$$

It is easy to see that $\rho(M_n) \geq \rho(M_2) > \mathcal{R}_{11}, \forall n > 2$. The inequality (11) holds.

Mobile WiMAX Network Security

Rainer Falk[1], Christian Günther[2], Dirk Kröselberg[2], and Avi Lior[3]

[1] Siemens Corporate Technology
`rainer.falk@siemens.com`
[2] Nokia Siemens Networks
`{christian.1.guenther,dirk.kroeselberg}@nsn.com`
[3] Bridgewater Systems
`avi@bridgewatersystems.com`

Abstract. WiMAX networks provide broadband data access to mobile as well as stationary users. While the wireless link is based on the 802.16e-2005 specification developed by IEEE, a complete network architecture "behind the base station" with global roaming support has been specified by the WiMAX forum. The security architecture for these networks covers EAP/AAA-based secure network access, secure bootstrapping of macro mobility based on Mobile IP, and secure over-the-air provisioning. Specific solutions have been standardized to support combined or separate device and user authentication.

1 Introduction

WiMAX is a technology providing mobile and stationary broadband wireless access to IP-based services through a common radio technology, providing support for quality-of-service, roaming of mobile users, and strong security. The wireless link has been specified by as IEEE 802.16e-2005 [1]. WiMAX comprises a wireless interface that is sometimes hyped as the next technology disruption after IEEE 802.11 wireless LAN. In reality, much more is covered than the pure radio interface. The WiMAX forum [2] has developed a complete network architecture [11] for 802.16-2005 wireless access that supports IP mobility and global roaming. It has similarities with cellular wireless standards such as 3GPP and 3GPP2 as well as with IEEE 802.11.

After giving an overview of the WiMAX network architecture in Section 2, its security architecture is described, highlighting the specific security solutions. Section 3 focuses on network access authentication, whereas Section 4 describes the WiMAX security and key management solution for mobility. Section 5 concludes with a summary and outlook.

2 WiMAX Network Architecture

The WiMAX network architecture complements the IEEE 802.16e-2005 wireless link with a network architecture partially based on IETF protocols. It supports mobility and global roaming, i.e. subscribers can get serviced by any visited operator that maintains a business and roaming relationship with their home operator. A mobile WiMAX-capable device is denoted as mobile station (MS). On the network side, WiMAX introduces two logical sets of network functions that can be mapped to corresponding business entities, or network operators, see Fig. 1.

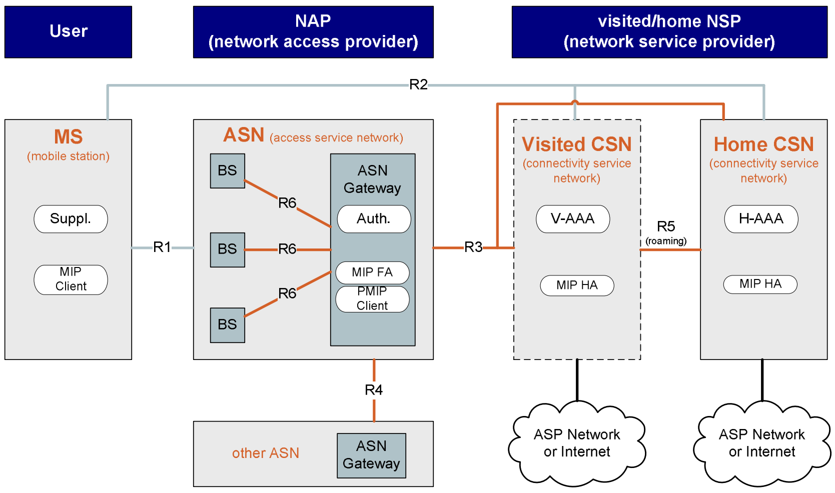


Fig. 1. WiMAX Network Architecture

- The access service network (ASN), which provides network access, comprises WiMAX base stations (BS) and introduces the ASN Gateway as a central controller. The ASN loosely corresponds to the radio access network of cellular systems.
- The connectivity service network (CSN), which corresponds rather to the core network of cellular systems, provides IP connectivity and comprises network entities like IP mobility anchor, DHCP server, AAA infrastructure, and subscriber management functions.

Reference points define the interfaces and group the different protocols between the entities of a WiMAX network. Some of these reference points connect different networks or business entities (shown at the top of Fig. 1). R3 connects ASN and CSN, or a network access provider (NAP) that is operating one or more ASNs to a network service provider (NSP) that is operating a CSN. Others, like R6, are internal to the ASN and connect one or more base stations to an ASN gateway. For roaming scenarios where a user gets WiMAX service through a different operator's network, two types of CSNs are involved, namely the visited (local) CSN and the home CSN the WiMAX user is subscribed to.

For protecting network access, access authentication in WiMAX builds on the IETF EAP (extensible authentication protocol) framework [3] and the RADIUS [8] or Diameter [10] protocol. It uses the common three-party EAP/AAA model for authentication and authorization, involving the supplicant (Suppl) on the mobile station MS, the authenticator (Auth) in the ASN, AAA proxies in the visited CSN (V-AAA) and the AAA server in the home CSN (H-AAA).

EAP itself provides a generic "container" and a well-defined state machine for carrying authentication protocols. Instances of authentication protocols are called

EAP methods. Depending on the EAP method, authentication can be unilateral, i.e. server-only authentication that is similar to common SSL or TLS usage in the Internet, or mutual. It can be based on either shared secrets (including passwords) or public/private key pairs and certificates. WiMAX supports cryptographic authentication of both the device and the subscriber (user) as one major difference to WLAN or mobile cellular network deployments. It defines, but is clearly not limited to, a set of default methods that comprise EAP-TLS [4], EAP-TTLS and EAP-AKA [5]. Any other EAP method matching WiMAX security requirements can be used when supported by the device and the CSN operator at the same time.

WiMAX defines two levels of mobility management, CSN anchored mobility that is based on Mobile IP, and ASN-anchored mobility:

- In the case of CSN anchored mobility or R3 mobility, the handoff takes place between administrative domains. Such R3 mobility can be handled by Mobile IP (MIP), see e.g. [9]. Two variants of MIP are supported: In the case of standard MIP (called CMIP or client-MIP in the WiMAX architecture), the MS itself performs mobility signaling, while in the case of proxy MIP (PMIP), a proxy mobile node (PMIP client) located within the ASN performs MIP signaling on behalf of the MS. So with PMIP, R3 mobility is possible even for MSs that do not support MIP. PMIP does not affect network access security, but has a certain impact on the required key distribution scheme, as – compared with CMIP – the PMIP client is part of a different entity having different security properties.
- Local handoffs in ASN-anchored mobility are handled by a single ASN gateway or between ASN gateways of the same administrative domain, without requiring any Mobile IP signaling towards the CSN. As the local handoff is realized on link layer, it does not affect the IP configuration of the MS.

3 Network Access Authentication

For fixed wireless access, IEEE 802.16-2004 [6] has a single network access authentication and authorization key establishment protocol Privacy Key Management (PKM). This protocol authenticates the MS using a public/private key pair. 802.16e-2005 [1] has adopted PKM for legacy reasons, but also introduced PKMv2 supporting EAP. Also, it allows for mutual authentication between MS and the backend AAA server instead of just the local BS.

For mobile WiMAX networks following the WiMAX Forum network specifications [11], security for the wireless link relies on the security measures provided by 802.16e-2005 [1]. These are profiled to limit the number of supported security options of the PKMv2 protocol, while not continuing support for the former PKM that is used by fixed WiMAX deployments. For authentication of WiMAX subscribers, mobile WiMAX networks build on EAP/AAA-based authentication. This allows for a large set of available EAP authentication methods to give each operator a maximum flexibility to match specific security needs. This section gives an overview of the mobile WiMAX network authentication.

3.1 Authentication Scenarios

For secure access to network resources a separation between authentication of the subscription and authentication of a mobile WiMAX device is introduced.

The former in the most common example maps to a human user. Subscription authentication typically ensures – from the operator's perspective – that the mobile user accessing network resources and mobile services can be uniquely and securely associated with a subscription. In contrast, different reasons may require authentication of the device itself, like for instance the protection of initial over-the-air subscription provisioning using long-term device credentials or verification that the device is not a stolen one.

Current examples of network access authentication are:

- *3GPP networks:* Subscription authentication is based on the tamper-resistant and removable SIM/USIM card that holds the subscription data and cryptographic keys. Device authentication happens only implicitly by verification of the mobile terminal's IMEI number. A USIM card and subscription can easily move between different different mobile terminals, so the subscriber's identity is not necessarily bound to any device identity.
- *WLAN networks:* Only a single EAP authentication is supported for WPA/WPA2 (Wi-Fi Protected Access). Authentication is performed using the EAP protocol towards an AAA server, or using a pre-shared key shared between WLAN devices, e.g. within a WLAN home network. Depending on the deployment, a user/subscriber or a device can be authenticated.
- *DSL or fixed WiMAX access [6]:* Here the device is authenticated. Subscription authentication is typically performed after the device is being connected, e.g. based on username/password credentials.

Device authentication becomes more important in mobile WiMAX networks. Off-the-shelf devices like standard notebooks - in contrast to today's mobile phones that are often sold and subsidized by cellular operators - cannot be expected to be under operator control. Hence, it makes much sense to introduce cryptographically strong device authentication based on certificates imprinted into the device or wireless interface card upon manufacturing. This will enable operators to control that only approved WiMAX equipment can access the operator's network resources and it especially will help to simplify the process of secure device provisioning. Another valid scenario for device authentication is based on the fact that in a roaming situation, the visited network may be subject to country-specific regulations that mandate cryptographic authentication of the device.

Certainly, the overall security level achieved by this depends on a number of challenging factors, like secure storage of keys and trusted certificates in the device, and the PKI (public-key infrastructure) system's responsibility to efficiently handle aspects like certificate revocation.

To cover all these scenarios, the WiMAX Forum defines means for performing both device and user authentication based on EAP methods, single or combined. Device certificates are pre-installed in all WiMAX-Forum compliant devices upon manufacturing, and a PKI including top-level certificate authorities to make those certificates work between different operators is made available through the WiMAX Forum. Detailed considerations related to this PKI system are provided by [12].

3.2 Subscription Authentication

Mobile WiMAX networks require secure access to their resources with authentication based on the EAP/AAA model. In principle, the same three-party authentication model of 802.1X access to WLAN networks is used.

There are a number of technical differences between WLAN and WiMAX like state machines or key derivation for the involved entities. Also, a major new aspect for mobile WiMAX is a central controller for a set of base stations which also handles authentication. With such a controller the Authenticator is not directly located in the base station any more, but is handled centrally by the ASN Gateway which may cover a large number of base stations at the same time. The base station itself only acts as a simple relay for EAP in this scenario, and is only provided with the keys required for protecting the wireless link after successful EAP authentication. One main advantage of this architecture is that during handover, the number of re-authentications can be reduced significantly, thereby avoiding delays.

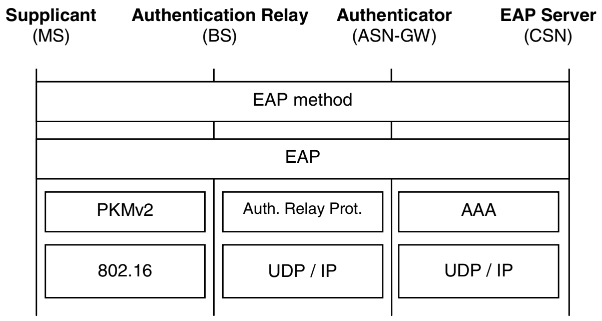


Fig. 2. WiMAX Protocol Layering for Network Access Authorization

An EAP method runs end-to-end between the supplicant in the MS and the EAP/AAA server. It is transparent to the Authenticator in the access network and to any intermediate RADIUS proxy or Diameter agent in the AAA path. Only MS and AAA server run the required security algorithms and possess the required security credentials. Hence, a decision on the EAP method to be used is up to the operator of the AAA server (as long as device vendors support the selected method). Subscription authentication in WiMAX always terminates in the AAA server of the subscriber's home network.

The roaming considerations shown in Fig. 1 also apply here. If the subscriber attaches to an ASN connected to a visited CSN operator, EAP authentication is routed based on the home domain indicated in the network access identifier NAI [7] provided by the MS. This passes through an AAA proxy in the visited CSN and is routed to the home CSN AAA server, where the subscription is actually authenticated.

The NAI for subscription authentication serves as the common identity for the EAP and Mobile IP session in the WiMAX system. It consists of a username part and a realm part identifying the subscriber's home network ("`username@realm`"). For supporting identity privacy, the NAI is allowed to carry a pseudonym as the username part instead of the real identity of the subscription. Hence, it is mainly used for AAA routing and binding different sessions related to the same network entry of the subscriber in the WiMAX network. This pseudo-identity can also be changed with each new authentication.

On the wireless link between MS and BS, EAP payloads are encapsulated in PKMv2 messages as defined by 802.16e-2005 [1] and are transferred over the 802.16 physical link (see Fig. 2). The authentication relay function (see section 4.4.2 of [11]) in the BS and the authenticator in the ASN-GW exchange EAP payloads by encapsulating them within WiMAX-specific protocol messages across the R6 reference point.

3.3 Device Authentication and Subscription Provisioning

WiMAX client devices are shipped with a device certificate that is securely imprinted by the device vendor upon manufacturing. These certificates are verified using a common root authority hosted by the WiMAX Forum [13]. This public-key infrastructure including root certificate authorities for both WiMAX server and device certificates enables certificate-based device authentication. Conceptually, the AAA server for device authentication is always located in the home CSN and in fact in most cases it will just be the same entity as the one being responsible for subscription authentication. However, the major use case for device authentication is currently to support the process of over-the-air self-provisioning of new subscriptions where the user looking for a WiMAX subscription does not yet have any “home” operator because no subscription has been established yet. This is especially important for devices like notebooks that are bought through common retail channels that are not related to any specific operator. In such cases, one of the WiMAX NSPs locally advertised through the radio channel during initial network attachment will be selected either manually by the user or automatically based on some policy preconfigured by the operator or device manufacturer in the MS. After successful initial provisioning of subscription data the selected NSP will become the home CSN operator for this new subscriber.

The limitation of terminating subscription and device authentication in the same operator’s network currently reduces complexity and therefore interoperability issues due to unclear policies between the potentially different involved ASN and CSN operators. Future extensions may support terminating device authentication also locally in the access network.

If both device and subscription authentication are to be performed for the same network entry, the technical choice is to use a tunneled EAP method with the default one being EAP-TTLS [14]. This makes use of the device certificate for setting up a TLS-based secure tunnel first and then runs the subscription authentication inside the tunnel. Options for the so-called “inner authentication” are either an MS-CHAPv2 based challenge-response authentication of the subscription credentials or any suitable EAP method that is forwarded through the outer TLS tunnel.

A different technical approach would have been to perform two subsequent but otherwise independent EAP protocol runs with cryptographically binding the two authentication exchanges in the access network (‘double-EAP’). Besides improved flexibility like support for the AAA servers being located in different locations and belonging to different operators or fewer limitations regarding the choice of EAP methods, this approach would have offered slightly advanced access security due to a cryptographic binding of the two authentication phases that is not available for the EAP-TTLS version currently used by the WiMAX Forum. However, due to a higher complexity in the wireless access this solution has been deprecated by both IEEE 802.16 and the WiMAX Forum NWG.

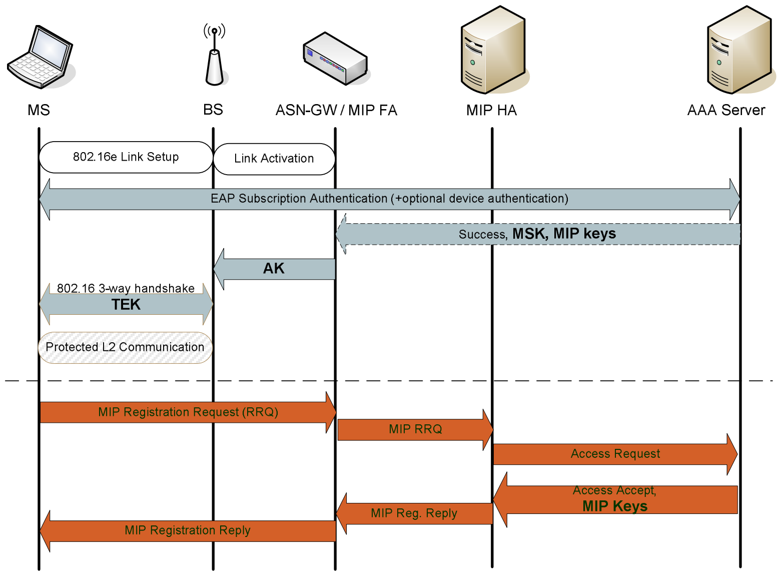


Fig. 3. WiMAX Network Access with Device and Subscriber Authentication

Figure 3 provides a high-level overview of the security-related steps that are performed in a mobile WiMAX network as part of successful network entry. Network access starts with activating the wireless link and selecting the desired NAP. The first EAP run after attaching to a BS of the NAP will be routed through the ASN's authenticator being in charge of this BS to the subscriber's home operator's AAA server. The latter is identified by the realm information of the NAI provided by the attaching MS. After successful EAP method termination the subscriber and the network are authenticated, and optionally the device itself is authenticated based on the device certificate. RADIUS or Diameter will deliver the resulting master session key MSK (see also Section 4) to the authenticator in the ASN Gateway (ASN-GW). A base station specific key AK and related security context information is derived and passed to the respective base station for securing the wireless link (see also Fig. 4 for the WiMAX key hierarchy). This key is used in the 802.16e-2005 3-way handshake authentication between MS and BS and to establish further traffic encryption keys (TEK). After successful authentication, the MS registers with its assigned Mobile IP home agent in case IP mobility service based on MIP is used. Mobility keys for protecting the Mobile IP signaling are also distributed during network access authentication (see Section 4).

Let us take a further look at device authentication in WiMAX network access for the purpose of setting up a new subscription, that is, the dynamic self-provisioning [15] after an un-provisioned device retrieves information about the locally advertised WiMAX CSN operators from the selected ASN and the user selects one for creating a new subscription. The MS enters the network using the EAP-TLS authentication method and the device certificate as the client-side security credential. Certificate-based server-authentication is also performed in EAP-TLS and a WiMAX device is expected to be shipped with pre-installed trusted root certificates to be able to verify the operator's AAA server certificate that is subject to the WiMAX Forum PKI.

After successful network entry, the user will start setting up the new subscription (typically by accessing a Web portal page) and the newly generated subscription data including security credentials and identity information has to be securely transmitted over-the-air to the WiMAX device.

This initial provisioning is based on specifications developed either by the Open Mobile Alliance (OMA) for management of mobile devices or by the Broadband Forum for CPE devices targeted for the fixed-line and DSL market. Both are extended and profiled by the WiMAX Forum [15] to cover the specifics of WiMAX devices. Security keys to enable the security mechanisms in both protocols between MS and provisioning server are derived from the EAP-TLS based device authentication procedure as subordinate keys generated from EMSK. While the MS is able to generate those keys internally, the provisioning server will receive them from the AAA server. So after network access authentication, a shared secret derived from the device authentication is established between the MS and the provisioning server to protect the communication for setting up the subscription.

4 Key Management for Mobility Support

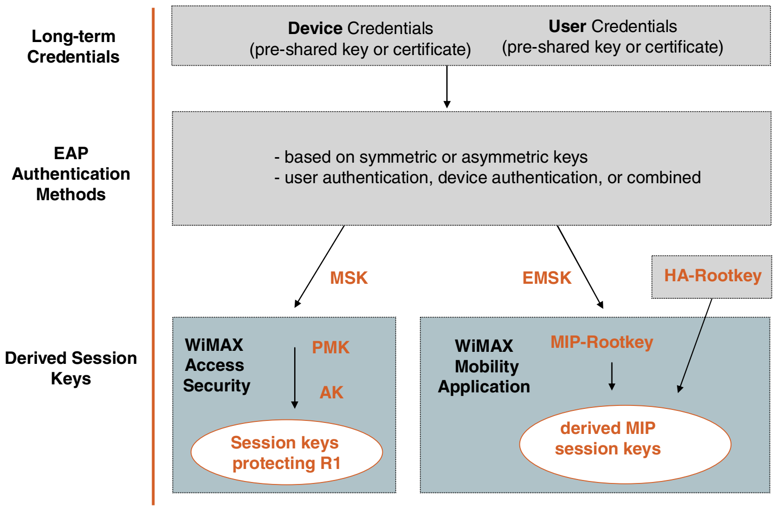
Two levels of mobility management are supported by WiMAX: CSN anchored mobility that is e.g. based on Mobile IP, and local ASN-anchored mobility. In the latter case, the traffic is redirected towards the new BS after hand-off, but the MIP Foreign Agent (FA) location or MS IP address do not change. In the case when the new BS is controlled by the same ASN Gateway as the previous BS, the authenticator in the same ASN Gateway is still in charge for securing the new wireless link after handover and will provide appropriate key material to the new BS over the WiMAX-defined R6 reference point. This key material is still derived from the original EAP authentication session key MSK. For hand-off to a base station controlled by a new ASN Gateway, the Authenticator in the new ASN Gateway may take over security control. This requires an EAP re-authentication to be performed and fresh keys to be generated to protect the new wireless link. Alternatively, the authenticator of the old ASN Gateway can be kept to avoid re-authentication and related delays. EAP exchanges go all the way between the MS and the home operator's AAA server. In this case it will remain the "anchor" authenticator and provide appropriate keys for the new BS to the new ASN Gateway, across the R4 interface.

When CSN-anchored mobility handover via Mobile IP takes place, the MS gets assigned a different local IP address after hand-off. Although both Mobile IP v4 and v6 can be supported, this section will focus on the Mobile IPv4 [9] aspects for the sake of brevity. The MS's local IP address called care-of-address (CoA) is registered with the MIP home agent (HA) that resides in the CSN (see Fig. 1). The HA handles the MS's long-term home IP address, which remains unchanged and tunnels data traffic destined to the MS to its current care-of address. The MS communicates using its home address. In WiMAX MIPv4, a foreign agent (FA) in the ASN gateway is the tunnel endpoint for all user traffic in the access network. WiMAX allows for a dynamic assignment of the HA and the MS' home address. The HA may, depending on operator policies, be assigned and located on a per-subscriber basis either in the home CSN or in a visited CSN. The network elements and interfaces relevant for Mobile IP and AAA interaction are shown in Fig. 1.

Two variants of MIP are supported: Either the MS performs MIP signaling itself (CMIP), or a PMIP client in the ASN-Gateway performs MIP signaling on behalf of the MS. As described above, it is possible to keep the anchor authenticator after handover. This means that the FA functionality moves to a new ASN Gateway, but no EAP re-authentication needs to be performed. The anchor authenticator is still in charge of providing keys for securing the wireless link. If PMIP is used in this scenario, the PMIP client also stays with the anchor authenticator and does not move to the new ASN Gateway.

Mobility signaling has to be protected, as it controls all the user traffic. Hence, Mobile IP requires a security association between the Mobile IP client and the HA (MN-HA) for protecting MIP signaling. Also, MIP signaling between FA and HA (FA-HA) is protected by a security association.

In WiMAX, the architectural approach was chosen to dynamically derive the mobility keys from the EAP-based network access authentication that has been described in Section 3. The resulting key hierarchy is illustrated in Fig. 4. Initially based on long-term credentials for secure authentication of the subscription for



Long-term Credentials
Device Credentials (pre-shared key or certificate)
User Credentials (pre-shared key or certificate)
EAP Authentication Methods
- based on symmetric or asymmetric keys
- user authentication, device authentication, or combined
MSK
EMSK
HA-Rootkey
Derived Session Keys
WiMAX Access Security
PMK
AK
Session keys protecting R1
WiMAX Mobility Application
MIP-Rootkey
derived MIP session keys

Fig. 4. The WiMAX Key Hierarchy

network entry, EAP authentication creates two session keys MSK (master session key) and EMSK (extended MSK) that are known by the EAP peer (client) and the AAA server. The EMSK key is not used for securing the wireless link and is reserved to serve as master key for derivation of further application-specific session keys. Such applications rather include those integral to the network architecture design and network operation instead of being general third-party applications that are typically independent of the WiMAX network.

The MSK key forms the basis for derivation of all session keys for protecting the 802.16e-2005 wireless link. It is generated as a result of the EAP method based authentication in the MS and the AAA server terminating EAP in the CSN. As highlighted in Fig. 3, it is delivered via the RADIUS or Diameter protocol from the AAA server to the authenticator in the ASN Gateway. Further information on the key hierarchy for the wireless link can be found in [1].

From the second key EMSK, a Mobile IP root key MIP-RK is generated in the AAA server and the EAP peer, and all user-related session keys for the MN-HA security association for a CMIP or PMIP session are derived from the root key. Although for PMIP such user-related keys would not be required (instead, a PMIP client could use the same MN-HA key for all active user sessions), CMIP and PMIP is actually using the same scheme in WiMAX. This allows to simplify the network signalling by only defining a single mechanism for keying the different R3 mobility schemes. In addition, the AAA server generates an independent root key HA-RK for protecting signaling between a specific HA and connected FAs.

The MIP session keys related to a specific user session are distributed to the authenticator in the ASN Gateway already as part of the RADIUS or Diameter message that the AAA server sends to the authenticator as shown in Fig. 3, to indicate successful authentication of the user. The HA also requires the MN-HA security association for protecting and verifying MIP messages. It requests keys, if not yet available, from the AAA server when receiving a MIP registration message. In contrast, the MS generates all keys on its own if CMIP is used. For PMIP, no Mobile IP related keys are required in the MS.

In the ASN, the authenticator is responsible for further distributing the appropriate MIP keys to the FA and the PMIP client for PMIP. This is especially the case for an FA being located in a different ASN Gateway. HA-RK is used in the authenticator and HA to derive fresh FA-HA keys. It is only distributed as required, and is not specific to a single MS or subscriber.

An important aspect from the security point-of-view is the secure binding between two otherwise independent procedures of authenticating network access and securing mobility service. WiMAX achieves this by using the NAI identity provided during network access authentication also in Mobile IP registration. This is sent by the HA together with a unique key identifier, the MIP security parameters index (MIP-SPI), to the AAA server and is subsequently used there to select the correct set of mobility keys. By this, the network ensures that the session created during network access authentication and the MIP session belong to the same user. For MIP-SPI creation, it is important to avoid collisions of SPI values for MN-HA security across the network. Such collisions would result in ambiguities for selecting the subscriber session's related MIP keys, leading to dropped sessions. Hence, WiMAX networks implement a specific procedure to ensure that such collisions cannot happen (see section 4.3.5 of [11]).

The WiMAX approach of binding key management and distribution for Mobile IP to the EAP-based network access authentication has been an important step for the WiMAX network security architecture, compared to any approach based on static pre-configuration of shared MIP keys between a subscriber's WiMAX device and the home network. Such binding, however, also creates a need for synchronization of key lifetimes between the different EAP and MIP sessions and correct removal of session state when the devices log off. In an environment supporting fully mobile devices and globally roaming users, the benefits like increased security through dynamically provisioned mobility keys and reduced administrative complexity justifies the effort.

5 Summary and Outlook

The overall goal of the WiMAX security architecture is to create an interoperable security solution that covers all relevant aspects of WiMAX networks based on commonly accepted security protocols like EAP and RADIUS/Diameter. One major objective is to minimize administrative effort and therefore reduce operational expenses by designing dynamic mechanisms for distributing configuration data in the system. Looking at security, full support for dynamic over-the-air provisioning and dynamic generation and distribution of security associations fall into this category. This holds for the provisioning based on device authentication as well as for network services like IP mobility.

WiMAX is designed with the clear goal of being extensible in the future. In particular, the security concepts are not strictly limited to the WiMAX environment, but can be expanded to leverage integration with other mobile communication technologies like those developed by 3GPP or 3GPP2. Inter-technology interworking has been an important factor in the design of the WiMAX security architecture and as one example the dynamic keying approach developed for Mobile IP is also followed by 3GPP in their specifications describing how to interwork with other access technologies like WiMAX.

The WiMAX security architecture comes with similarities to WLAN (EAP) and 3GPP2 (Mobile-IP/AAA), but also with a number of new approaches: For network access, it supports EAP-based authentication of both the user's device and the subscription. Integral support for over-the-air provisioning combined with dynamic network selection support allow users of any type of WiMAX device to flexibly create a subscription with one of the available network service providers. Mobile IP security is dynamically bootstrapped. Furthermore, the real user identity can be hidden from the wireless link and from local access networks by using a pseudo-identity mechanism.

Certainly, the WiMAX network specifications are evolving, and ongoing work related to the next releases of the WiMAX network architecture will also further extend the security architecture. Besides obvious and already available enhancements like including full support for Diameter [10] as AAA protocol in addition to the initial RADIUS-only specifications, or support for WiMAX-specific SIM cards, improvements like a cryptographically protected rejection cause indication in cases where a WiMAX device is denied network entry, or advanced access scenarios like the support for Femto cells are being worked on in the WiMAX Forum.

References

1. IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, IEEE 802.16e-2005 and 802.16/COR1, `http://www.ieee802.org/16/`
2. WiMAX Forum, `http://www.wimaxforum.org/`
3. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP), RFC 3748 (June 2004), `http://www.ietf.org/rfc/rfc3748.txt`
4. Aboba, B., Simon, D.: PPP EAP TLS Authentication Protocol, RFC 2716 (October 1999), `http://www.ietf.org/rfc/rfc2716.txt`
5. Arkko, J., Haverinen, H.: Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA), RFC 4187 (January 2006), `http://www.ietf.org/rfc/rfc4187.txt`
6. IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, IEEE 802.16-2004, `http://www.ieee802.org/16/`
7. Adoba, B., Beadless, M., Arkko, J., Eronen, P.: The Network Access Identifier, RFC 4282 (December 2005), `http://www.ietf.org/rfc/rfc4282.txt`
8. Rigney, C., Willens, S., Rubens, A., Simpson, W.: Remote Authentication DiaI In User Service (RADIUS) (June 2000), `http://www.ietf.org/rfc/rfc2865.txt`
9. Perkins, C. (ed.): IP Mobility Support for IPv4, RFC 3344 (August 2002), `http://www.ietf.org/rfc/rfc3344.txt`
10. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter Base Protocol, RFC 3588 (September 2003), `http://www.ietf.org/rfc/rfc3588.txt`
11. WiMAX Forum: WiMAX Forum Network Architecture (Stage-3: Detailed Protocols and Procedures), Release 1 Version 4, (February 2009), `http://www.wimaxforum.org/sites/wimaxforum.org/files/documentation/2009/WMF-T33-001-R010v04_Network-Stage3-Base.pdf`
12. WiMAX Forum: WiMAX Forum Network Architecture – WiMAX Forum Device PKI Certificate Policy Draft Specification, Version 1.0.3 (April 2008), `http://www.wimaxforum.org/certification/x509_certificates/pdfs/wimax_forum_device_public_key_infrastructure_certificate_policy.pdf`
13. WiMAX Forum: X.509 ordering process, `http://www.wimaxforum.org/certification/x509_certificates/`
14. Funk, P., Blake-Wilson, S.: Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0), RFC5281 (August 2008), `http://www.ietf.org/rfc/rfc5281.txt`
15. WiMAX Forum: WiMAX Forum Network Architecture - WiMAX Over-The-Air General Provisioning System Specifications, Release 1.5 Version 1.0.2 (September 2008)

LoPSiL: A Location-Based Policy-Specification Language

Jay Ligatti, Billy Rickey[⋆], and Nalin Saigal

Department of Computer Science and Engineering
University of South Florida
{ligatti,brickey,nsaigal}@cse.usf.edu

Abstract. This paper describes the design of LoPSiL, a language for specifying location-dependent security and privacy policies. Policy-specification languages like LoPSiL are domain-specific programming languages intended to simplify the tasks of specifying and enforcing sound security policies on untrusted (i.e., potentially insecure) software. As far as we are aware, LoPSiL is the first imperative policy-specification language to provide abstractions specifically tailored to location-dependent policies for mobile-device applications. We have implemented a proof-of-concept compiler that inputs a LoPSiL policy P and a mobile-device application program A and outputs a new application program A' equivalent to A, except that A' contains inlined enforcement code that ensures that A' satisfies P at runtime. We report our experiences using this compiler to design and implement several policies for mobile-device applications.

Keywords: Policy-specification languages, location-dependent policies, mobile devices, security and privacy.

1 Introduction

Policy-specification languages are domain-specific programming languages intended to simplify the tasks of specifying and enforcing sound security policies on untrusted (i.e., potentially insecure) software. There are two common motivations for using policy-specification languages:

1. Users often wish to download and execute third-party software applications but do not (and should not) trust that those applications will behave securely or respect their privacy. Therefore, users (or other parties working on behalf of the users, such as vendors or system administrators) may utilize policy-specification languages to create and enforce customized, flexible constraints (i.e., policies) on the untrusted software. For example, a policy-specification language may be used to specify and enforce that applications downloaded from third-party providers do not delete particular files or read particular regions of memory.

[⋆] Rickey participated on this project as an NSF-REU (National Science Foundation Research Experience for Undergraduates) student at USF in the summer of 2007.

2. Application developers often wish to enforce security policies on their own code. Typically, application developers implement security and privacy considerations by scattering security checks throughout their code. Policy-specification languages and compilers enable those developers to refactor their code by moving all the scattered security checks from the application implementation and into a separate, isolated policy module (written in a language designed to make it easy to specify the desired policy). Separating the security policy from the core application code provides application developers all the standard software-engineering benefits one would expect from modularization: it makes the centralized policy easier to create, locate, analyze, and maintain.

A rich variety of expressive (i.e., imperative and Turing-complete) policy-specification languages and systems has been implemented [16,10,9,11,12,21,5,13]. All of these languages enable users to centrally specify security and privacy policies to be enforced on untrusted software at runtime. All of the cited languages have also been implemented as compilers that convert untrusted into trustworthy applications by inputting a policy P and an application program A and outputting a new application program A' equivalent to A, except that A' contains inlined enforcement code that ensures that A' satisfies P at runtime.

However, as far as we are aware, no imperative policy-specification languages have yet targeted location-dependent policies for securing applications on mobile devices (e.g., roaming laptops, cell phones, robots, PDAs, etc)—though researchers have previously developed declarative (i.e., Turing-incomplete) policy-specification languages limited to location-based access-control policies [7,2,1].

The lack of expressive policy-specification languages for mobile-device applications presents a problem—and an opportunity—because:

– The two motivations for policy-specification languages given above are becoming more relevant for mobile-device applications as mobile devices become more powerful, open, and flexible and allow users to download and execute third-party software.
– Security and privacy policies for applications on mobile devices often have to reason about different machine states—*locations* in particular—than policies for applications on immobile devices.

Hence, it is important to consider how to create a convenient policy-specification language specifically for mobile-device applications. This paper proposes such a language, called LoPSiL, whose primary novelty is to provide several abstractions for conveniently accessing and manipulating location information in policy specifications.

Roadmap. We proceed as follows. Section 2 describes the design of LoPSiL, its core constructs for simplifying the specification of location-dependent policies in Section 2.1 and several examples highlighting its ease of use in Section 2.2. Section 3 discusses our proof-of-concept implementation of a LoPSiL compiler and experiences we have had designing and implementing LoPSiL policies. Section 4 concludes and describes possible future work.

2 LoPSiL

Due to the popularity of Java, particularly Java ME, as an application programming language for mobile devices [15], we have chosen to design and implement LoPSiL constructs in Java source code. Also, to make it easy for security engineers to learn and use LoPSiL, and to simplify the implementation of a LoPSiL compiler, we have packaged LoPSiL as a Java library, to which LoPSiL policies may refer (e.g., a LoPSiL policy may refer to the `Location` class in the LoPSiL library). Although we treat LoPSiL in a Java context in this paper, we have built LoPSiL on six core abstractions that are application-language independent, so we expect LoPSiL to be portable to other languages and platforms.

2.1 Core Linguistic Constructs

LoPSiL is built on six core abstractions; we describe each in turn.

Locations. In LoPSiL, `Location`s are (possibly abstract) places. They may refer to rooms, chairs, floors, buildings, campuses, GPS coordinates, regions of a network topology, etc. All `Location`s have an identity (e.g., a room or building name, or coordinates in GPS). LoPSiL provides many built-in utility methods for manipulating GPS locations (e.g., to calculate distances between them), as the examples in Section 2.2 demonstrate. However, LoPSiL users are always free to implement custom methods for manipulating locations (e.g., to define a containment relation over locations, useful for testing whether a room is in a building, a building is on a campus, etc).

LocationDevices. A `LocationDevice` is LoPSiL's interface to real-time location information. Concrete `LocationDevice`s must implement two abstract methods. The first simply informs policies of the device's current location, which could be determined using GPS or by inputting location information from a user, a file, another (networked) host, a TLTA device [22], etc. The second abstract method `LocationDevice`s must implement informs LoPSiL policies of the device's *granularity*, that is, with what precision is the device's location information accurate (e.g., accurate within 1 meter, 1 room, 1 road, 1 building, 1 kilometer, etc). LoPSiL policies can require devices to provide location information with particular granularity thresholds.

Our LoPSiL implementation includes concrete implementations of two `LocationDevice`s, but users are always free to implement others. The first `LocationDevice` provided with LoPSiL represents and connects to a Garmin GPS device using Java's communication API and the GPSLib4J library [14]; the second `LocationDevice` represents and connects to a simple GUI with which users can manually select their current location from a list of known locations.

PolicyAssumptions. LoPSiL policies may make two important assumptions about `LocationDevice`s. First, as mentioned above, a policy may require location information with a particular granularity (e.g., accurate within 15m). Second, a policy may require that location updates arrive with a particular frequency (e.g., a new update must arrive within 10s of the previous update). LoPSiL policies encapsulate these assumptions, along with the `LocationDevice`s whose location data they trust, in a `PolicyAssumptions` object. A LoPSiL policy gets notified automatically whenever a `LocationDevice` violates the policy's granularity or frequency-of-updates assumptions.

Actions. An `Action` encapsulates information about a *security-relevant method* (i.e., any Java application or library method of relevance to a LoPSiL policy). LoPSiL policies can interpose before and after any security-relevant action executes; the policy specification then determines whether that action is allowed to execute. Policies may analyze `Action` objects to determine which security-relevant method the action represents, that method's signature, run-time arguments, and calling object (if one exists), whether the method is about to execute or has just finished executing, and the return value of the action if it has finished executing.

Reactions. LoPSiL policies convey decisions about whether to allow security-relevant `Action`s to execute by returning, for every `Action` object, a `Reaction` object. An *OK reaction* indicates that the action is safe to execute; an *exception reaction* indicates that the action is unsafe, so an exception should be raised (which the application may catch) instead of allowing the method to execute; a *replace reaction* indicates that the action is unsafe, so a precomputed return value should be returned to the application in place of executing the unsafe action; and a *halt reaction* indicates that the action is unsafe, so the application program should be halted.

Policies. LoPSiL policies incorporate all of the previously described language constructs. There are five parts to a LoPSiL `Policy` object:

1. A policy may declare `PolicyAssumptions` upon which it relies.
2. A policy may define a `handleGranularityViolation` method, which will be invoked whenever all `LocationDevice`s upon which the policy relies violate the policy's location-granularity assumption.
3. A policy may define a `handleFrequencyViolation` method, which will be invoked whenever all `LocationDevice`s upon which the policy relies violate the policy's frequency-of-update assumption. LoPSiL's `PolicyAssumptions` class implements the multithreading needed to test for frequency-of-update violations.
4. A policy may define an `onLocationUpdate` method, which will be executed any time any `LocationDevice` associated with the policy updates its `Location` information. This method enables a policy to update its security state and take other actions as location updates occur in real time.
5. A policy must define a `react` method to indicate how to react to any security-relevant method. LoPSiL requires every policy to contain a `react` method, rather than providing a default allow-all `react` method; hence, policy authors wanting to allow all security-relevant methods to execute unconditionally must explicitly specify their policy to do so.

```
public class AllowAll extends Policy {
  public LocationDevice[] devices = {new LopsilGPS(LopsilGPS.GARMIN)};
  public LocationGranularityAssumption lga =
    new LocationGranularityAssumption(15, Units.METERS);
  public FrequencyOfUpdatesAssumption foua =
    new FrequencyOfUpdatesAssumption(10, Units.SECONDS);
  public PolicyAssumptions pa =
    new PolicyAssumptions(this, devices, lga, foua);
  public void handleGranularityViolation() {System.exit(1);}
  public void handleFrequencyViolation() {System.exit(1);}
  public synchronized void onLocationUpdate() {
    System.out.println("new location = " + devices[0].getLocation());
  }
  public synchronized Reaction react(Action a) {
    return new Reaction("ok");
  }
}
```

Fig. 1. Simple LoPSiL policy that prints location information as it is updated and allows all security-relevant methods to execute as long as its location-granularity and frequency-of-update assumptions are not violated

Figure 1 contains a simple LoPSiL policy with all five of these components.

Existing policy-specification languages, such as Naccio [12], PSLang [11], and Polymer [5,4], provide constructs similar to our `Actions`, `Reactions`, and `Policy` modules with `react`-style methods. LoPSiL's novelty is its addition of optional location-related policy components: `Locations`, `LocationDevices`, granularity and frequency-of-update assumptions, and methods to handle granularity and frequency-of-update violations and to take action when location state gets updated (with the `onLocationUpdate` method).

2.2 Example Policies

We next survey four location-dependent runtime policies and show how to specify them in LoPSiL. The first is an example of the sort of policy a user might wish to enforce on untrusted third-party software, while the other three are examples of policies that application developers might wish to enforce on their own software. We have enforced and tested versions of all these example policies on Java applications executing on a roaming laptop.

Access-Control Policy.[1] Our first example is a privacy-based access-control policy that constrains an application's ability to read location data at particular times. The policy, shown in Figure 2, requires that monitored applications can only access the device's GPS data from 08:00 (8am) to 18:00 (6pm) on workdays.

[1] We thank Sean Barbeau at USF's Center for Urban Transportation Research for suggesting this policy.

```
public class NoGpsOutsideWorkTime extends Policy {
  public synchronized Reaction react(Action a) {
    if(ActionPatterns.matchesGpsRead(a) && !TimeUtils.isWorkTime())
      //return a null location to the application
      return new Reaction("replace", null);
    else return new Reaction("ok");
  }
}
```

Fig. 2. LoPSiL policy preventing an application from reading GPS data outside of work hours

```
public class ShowNavigation extends Policy {
  public LocationDevice[] devices = {new LopsilGPS(LopsilGPS.GARMIN)};
  public PolicyAssumptions pa =
  ...
  public synchronized void onLocationUpdate() {
    if(devices[0].getLocation().
         distance(getExpectedCurrentLocation(), Units.METERS)>10)
      AppGUI.displayNavigationalAid();
  }
}
```

Fig. 3. Abbreviated LoPSiL policy requiring that navigational aid appear when the device's current location deviates from its expected path

A user might want to enforce such a policy to prevent an employer-provided application from learning the device's location when the employee is not at work (e.g., so the employer does not know where the employee shops, or how much time the employee spends in certain places during the employee's off hours). In fact, providing for the enforcement of such a policy might be the only way the employer could convince the employee to run a work-related application on the employee's mobile device.

Deviation-from-path Policy. Our second example policy requires navigational aid to appear when the device's location deviates more than 10m off its expected path. The policy code, shown in Figure 3, invokes a method called `getExpectedCurrentLocation` to determine where the policy currently expects the device to be. Method `getExpectedCurrentLocation` could return a location based on the route being displayed to the user (as in dashboard-mounted GPS systems), on traffic conditions, on the path the user normally travels in this area, etc.

Safe-region Policy.[2] Another interesting sort of policy expressible in LoPSiL is shown in Figure 4. This policy, intended to monitor software on a robot, requires

[2] We thank Robin Murphy at Texas A&M University for suggesting this policy.

```
public class SafeRegion extends Policy {
  private Location[] safeRegionEndpoints;
  private boolean inRegion;
  public SafeRegion() {
    safeRegionEndpoints = getSafeRegionLocs();
    inRegion=devices[0].getLocation().inRegion(safeRegionEndpoints);
  }
  public PolicyAssumptions pa = ...
  public synchronized void onLocationUpdate() {
    inRegion=devices[0].getLocation().inRegion(safeRegionEndpoints);
  }
  public synchronized Reaction react(Action a) {
    if(!inRegion && ActionPatterns.matchesPlainWrite(a)) {
      String encMsg = encrypt(a.getArgs()[0].toString());
      try { //to replace the unencrypted send with an encrypted send
        ((BufferedWriter)(a.getCaller())).write(encMsg);
      } catch(IOException e) {...}
      return new Reaction("replace", null);
    } else return new Reaction("ok");
  }
}
```

Fig. 4. Abbreviated LoPSiL policy requiring robot-control software to encrypt outgoing messages when the robot is outside a secure-region perimeter

the robot to encrypt all outgoing communications when the robot's location is outside a secure-region perimeter.

Social-networking Policy. Our final example is a social-networking policy in which the user's friends get invited to rendezvous when the user travels to a new area. Specifically, the policy requires that if:

– the device has traveled more than 100km over the past 2 hours (i.e., average speed has been more than 50km/hr),
– the device has traveled less than 2km over the past 20 minutes (implying that the user's travels have at least temporarily ended), and
– the policy enforcer has not sent invitations to friends in the past hour,

then the policy enforcer must:

– broadcast a "Where are you?" message to all friends in the user's address book,
– collect responses from the friends, and
– send invitations to meet to those friends now within 20km of the user.

An abbreviated LoPSiL policy specifying such constraints appears in Figure 5.

```
public class InviteFriendsInNewArea extends Policy {
  //maintain a buffer of two hours' worth of location data
  private LocBuffer longBuf = new LocBuffer(2, Units.HOURS);
  //maintain another buffer of twenty minutes' worth of location data
  private LocBuffer shortBuf = new LocBuffer(20, Units.MINUTES);
  private Time timeLastInvited = Time.NEVER;
  public PolicyAssumptions pa = ...
  public synchronized void onLocationUpdate() {
    Location currentLoc = devices[0].getLocation();
    longBuf.add(currentLoc);
    shortBuf.add(currentLoc);
    if(longBuf.earliest().distance(currentLoc, Units.KILOMETERS)>100
      && shortBuf.earliest().distance(currentLoc, Units.KILOMETERS)<2
      && timeLastInvited.elapsed(Time.getCurrentTime(),Units.HOURS)>1)
    {
      Location[] friendLocs = getFriendLocations();
      inviteLocalFriends(friendLocs,currentLoc,20,Units.KILOMETERS);
      timeLastInvited = Time.getCurrentTime();
    }
  }
}
```

Fig. 5. A location-dependent social-networking policy specified in LoPSiL

3 A LoPSiL Compiler

This section describes our implementation of LoPSiL and briefly reports on our experiences designing and implementing LoPSiL policies. The implementations of LoPSiL's basic `Location`, `LocationDevice`, `Action`, `Reaction`, and `Policy` modules occupy 1588 lines of Java code, while the implementations of our `GarminGpsDevice` and `LopsilWindowDevice` respectively occupy 847 and 107 lines of Java code. Our implementation is available online [20].

3.1 Compiler Architecture

A LoPSiL compiler needs to input a LoPSiL policy and an untrusted application, build a trustworthy application by inserting code into the untrusted application to enforce the input policy, and then output the trustworthy application. The standard technique for implementing such a compiler involves inlining policy code into the untrusted application. Several tools exist for inlining code into an application; a convenient tool for our purposes is an AspectJ compiler [3]. AspectJ compilers inline calls to *advice* at control-flow points specified by *point cuts* [17]. In the domain of runtime policy enforcement, *advice* refers to policy-enforcement code and *point cuts* to the set of security-relevant methods. We wish to interpose and allow policy-enforcement code to execute before and after any security-relevant method invoked by the untrusted application.

```
void java.io.PrintStream.println(..)
* javax.swing.JOptionPane.*(..)
java.util.Date.new()
```

Fig. 6. Example `.srm` file indicating that the accompanying LoPSiL policy considers security relevant all void-returning `java.io.PrintStream.println` methods, all methods in the `javax.swing.JOptionPane` class, and the parameterless constructor for `java.util.Dates`

LoPSiL users convert an untrusted application into a trustworthy application as follows.

1. The user creates a specification of the desired policy in a `.lopsil` file.
2. The user also creates a listing of all the methods the desired LoPSiL policy considers security relevant. This listing indicates to the compiler which application and library methods it needs to insert policy-enforcement code around. Policies get to interpose and decide whether (and how) all security-relevant methods may execute. The listing of security-relevant methods goes into a `.srm` file, one method signature per line. Figure 6 contains an example and illustrates how wildcards can be used in `.srm` files.
3. The LoPSiL compiler inputs the policy (`.lopsil`) and security-relevant-methods (`.srm`) into a `lopsil2aj` converter, which converts LoPSiL code into AspectJ code. The converter, implemented in 201 lines of Java, begins by converting the LoPSiL policy to Java source (in a `.java` file) by simply inserting three lines of code to import LoPSiL-library classes into the policy. The converter then creates an AspectJ-code file (`.aj`) that defines two things. First, the AspectJ code defines a point cut based on the declared security-relevant methods. Second, the AspectJ code defines advice to be executed whenever the point cut gets triggered (i.e., before and after any security-relevant method executes). This advice builds an `Action` object to represent the invoked security-relevant method, passes that `Action` to the LoPSiL policy (now in a `.java` file), obtains the policy's `Reaction` to the `Action`, and guides execution appropriately based on that `Reaction`.
4. Finally, the LoPSiL compiler inputs the untrusted mobile-device application (comprised of a set of `.class` files) and the `.java` and `.aj` files created in Step 3 into a standard AspectJ compiler [3]. The AspectJ compiler inlines the advice into the application before and after all security-relevant methods, thus producing an application that is secure with respect to the original LoPSiL policy.

Figure 7 presents an overview of this architecture.

Because LoPSiL uses AspectJ as its application rewriter, LoPSiL inherits AspectJ's limitations. Most importantly, the AspectJ compiler cannot rewrite (i.e., inline code into) methods in standard Java libraries; it can only rewrite application files. Therefore, our LoPSiL compiler can only ensure that policy-enforcement code executes before and after security-relevant methods invoked

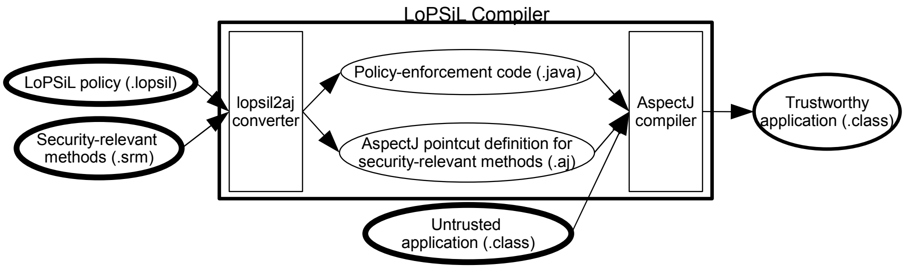

LoPSiL Compiler
LoPSiL policy (.lopsil)
Security-relevant methods (.srm)
lopsil2aj converter
Policy-enforcement code (.java)
AspectJ pointcut definition for security-relevant methods (.aj)
AspectJ compiler
Trustworthy application (.class)
Untrusted application (.class)

Fig. 7. Overview of the LoPSiL compiler. The compiler inputs `.lopsil`, `.srm`, and `.class` files and outputs the same `.class` files but with policy-enforcement code inlined before and after all security-relevant methods.

by the application being monitored. The important consequence is that our implementation does not allow enforcement mechanisms to interpose and make decisions about the execution of library methods invoked by other library methods. We could circumvent this limitation by writing our own LoPSiL enforcement-code inliner (e.g., using tools like the Bytecode Engineering Library [8]), as previous work has done [11,5,4], but at the price of significantly increased implementation complexity.

3.2 Experiential Observations

Having implemented the example policies described in Section 2.2, we believe that the six core constructs underlying LoPSiL serve as good abstractions for specifying location-dependent runtime security policies. This belief stems from the fact that LoPSiL was sufficiently expressive for us to specify every location-dependent policy we considered enforcing. In addition, after implementing LoPSiL, none of the example policies from Section 2.2 took us more than 2 hours to design, specify, and test. Although these results are encouraging, we need more experience with LoPSiL, and feedback from other users, before we can more completely and objectively evaluate its expressiveness and ease of use.

Another interesting outcome of designing the example policies from Section 2.2 is that we have observed some common, recurring uses of location information in security and privacy policies. Our location-dependent policies consistently based policy decisions on:

– The current absolute location of the device (e.g., whether the device is in the user's office)
– The geographic relationship of the device's current location with another location (e.g., whether the device is north of or within 1km of another location)
– The geographic relationship of the device's current location with a region of locations (e.g., whether the device is in an area of trusted terrain or within 10m of an expected path)
– The velocity or acceleration of the device

Because location-dependent policies consistently use location information in these ways, we provide several utility methods in LoPSiL for calculating distances, boundaries, velocities, and accelerations between locations. All policies can access these utility methods (cf. Figures 3–5) and can define custom operators on locations when the built-in methods are insufficient.

We have focused our efforts to date on LoPSiL's design, rather than the performance of our proof-of-concept compiler. Although we have not measured the performance overhead induced by LoPSiL-policy enforcement, we refer to previous work to argue that this performance concern is minor in practice when the application code runs on general-purpose, high-power, mobile machines (e.g., laptops), unless the policy itself specifies expensive computations or considers frequently invoked methods to be security relevant [12,11,5]. In the future, we would like to explore the performance impact of enforcement systems like LoPSiL on smaller and less powerful mobile devices.

4 Conclusions and Future Work

We have presented LoPSiL, a language for specifying location-dependent runtime security policies. LoPSiL's novelties are its abstractions for accessing and reasoning about location information in imperative policy specifications. In our preliminary experiments specifying policies for applications on a mobile laptop, we found these abstractions expressive and convenient. Given the increasing ubiquity of mobile devices, and the unique abstractions needed to conveniently deal with location information in security policies, we believe the research area of location-based policy-specification languages will for some time be an important topic of consideration for the programming-languages, computer-security, and mobile-device-applications research communities.

There are many opportunities for future work. One unresolved question is: how tolerable are the performance degradations that result from enforcing application-level runtime policies on weakly powered, inexpensive, and/or computationally constrained mobile devices? As Section 3.2 mentioned, for simplicity we have only enforced LoPSiL policies on a powerful roaming laptop. In the future we would like to perform a larger case study on less powerful mobile devices, such as cell phones; this would provide a more realistic measurement of enforcement overhead and give us more experience programming in LoPSiL.

It would also be interesting to investigate how to incorporate technologies, such as sophisticated static analyses [6] or policy structures [5], to simplify the task of specifying complex policies in LoPSiL. Such an investigation might lead to technologies for specifying complex location-dependent policies more conveniently as compositions of simpler subpolicy modules.

Finally, previous work has shown that policy-specification languages like LoPSiL, which allow monitoring mechanisms to store a complete trace of the application-program execution being monitored, enable specification and enforcement of all safety policies (such as the access-control policy in Figure 2) and some liveness policies (such as the social-networking policy in Figure 5) [18,19].

In the future we would like to include location information, mobile applications, and mobile monitoring mechanisms in formal models of policy enforcement, in order to better understand the precise space of policies enforceable with LoPSiL.

Acknowledgments. This research was supported in part by ARI grant W74V8H-05-C-0052 and by National Science Foundation Grants IIS-0453463, CNS-0831785, CNS-0742736, and CNS-0716343. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. Anisetti, M., Ardagna, C., Bellandi, V., Damiani, E.: Openambient: A Pervasive Access Control Architecture. In: Schmidt, A., Kreutzer, M., Accorsi, R. (eds.) Long-Term and Dynamical Aspects of Information Security: Emerging Trends in Information and Communication Security. Nova Science Publisher, Bombay (2007)
2. Ardagna, C., Cremonini, M., Damiani, E., di Vimercati, S., Samarati, P.: Supporting Location-based Conditions in Access Control Policies. In: Symposium on Information, Computer and Communications Security (2006)
3. The AspectJ Project, `http://www.eclipse.org/aspectj/`
4. Bauer, L., Ligatti, J., Walker, D.: Composing Expressive Run-time Security Policies. ACM Transactions on Software Engineering and Methodology (to appear)
5. Bauer, L., Ligatti, J., Walker, D.: Composing Security Policies with Polymer. In: ACM Conference on Programming Language Design and Implementation (2005)
6. Bauer, L., Ligatti, J., Walker, D.: Types and Effects for Non-interfering Program Monitors. In: Okada, M., Pierce, B., Scedrov, A., Tokuda, H., Yonezawa, A. (eds.) Software Security—Theories and Systems. Springer, Heidelberg (2003)
7. Bhatti, R., Damiani, M., Bettis, D., Bertino, E.: Policy Mapper: Administering Location-based Access-control Policies. IEEE Internet Computing 12(2), 38–45 (2008)
8. Byte Code Engineering Library, `http://jakarta.apache.org/bcel/`
9. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–39. Springer, Heidelberg (2001)
10. Edjlali, G., Acharya, A., Chaudhary, V.: History-based Access Control for Mobile Code. In: ACM Conference on Computer and Communications Security (1998)
11. Erlingsson, Ú., Schneider, F.: IRM Enforcement of Java Stack Inspection. In: IEEE Symposium on Security and Privacy (2000)
12. Evans, D., Twyman, A.: Flexible Policy-directed Code Safety. In: IEEE Symposium on Security and Privacy (1999)
13. eXtensible Access Control Markup Language (XACML) version 2.0, `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`
14. GPSLib4J v0.1, `http://gpslib4j.sourceforge.net/`
15. The Java ME Platform - the Most Ubiquitous Application Platform for Mobile Devices, `http://java.sun.com/javame/index.jsp`
16. Jeffery, C., Zhou, W., Templer, K., Brazell, M.: A Lightweight Architecture for Program Execution Monitoring. In: Program Analysis for Software Tools and Engineering (PASTE), pp. 67–74. ACM Press, New York (1998)

17. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.: An Overview of AspectJ. In: Knudsen, J.L. (ed.) ECOOP 2001. LNCS, vol. 2072, p. 327. Springer, Heidelberg (2001)
18. Ligatti, J., Bauer, L., Walker, D.: Enforcing Non-safety Security Policies with Program Monitors. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 355–373. Springer, Heidelberg (2005)
19. Ligatti, J., Bauer, L., Walker, D.: Run-time Enforcement of Nonsafety Policies. ACM Transactions on Information and System Security 12(3), 1–41 (2009)
20. LoPSiL Implementation, `http://www.cse.usf.edu/~ligatti/projects/runtime/LoPSiL.zip`
21. Robinson, W.: Monitoring Software Requirements Using Instrumented Code. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, p. 276.2 (2002)
22. Schmidt, A., Kuntze, N., Abendroth, J.: Trust for Location-based Authorisation. In: Wireless Communications and Networking Conference, pp. 3163–3168 (2008)

Impersonation Attacks on a Mobile Security Protocol for End-to-End Communications

Reiner Dojen, Vladimir Pasca, and Tom Coffey

Data Communications Security Laboratory,
Department of Electronic & Computer Engineering,
University of Limerick,
Limerick, Ireland
{reiner.dojen,vladimir.pasca,tom.coffey}@ul.ie

Abstract. This paper presents an analysis of a cryptographic security protocol that is designed for use in a mobile communication environment. The goal of the analysed protocol is to ensure secure end-to-end communication between two mobile users that are connected to different base stations. The analysis reveals a serious flaw in the used signature scheme of the security protocol. Exploitation of this flaw enables an intruder to use algebraic simplifications to forge signatures on arbitrary messages. Two attacks, which exploit this weakness, are detailed showing the impersonation of a mobile user and a base station, respectively. Corrections to the flawed protocol are proposed and analysed. It is established that the corrected protocol is secure against the presented attacks.

Keywords: Mobile end-to-end communication, analysis of security protocols, impersonation attack, authentication and secrecy protocol.

1 Introduction

Wireless communications are driven by the need to provide network access for mobile computing devices. At the same time, the increase in available online services causes users to transmit more and more sensitive information. Security protocols are used to protect this sensitive information against many forms of attack. Basic security protocols allow agents to authenticate each other, to establish fresh session keys for confidential communication and to ensure the authenticity of data and services. Building on such basic security protocols more advanced services like non-repudiation, electronic payment and electronic contract signing are achieved. However, the design of effective security protocols is a challenging problem and often weaknesses are discovered in published protocols [1], [2], [3]. Additionally, mobile communication systems have some unique difficulties, such as limited bandwidth, high latency and unstable connections. Further, mobile devices often have low computational and storage capacities. These difficulties seriously limit the choices of applicable techniques to provide suitable protection. Thus, many security protocols have been specially designed for mobile communications [4], [5], [6], [7], [8].

This paper presents the analysis of the mobile end-to-end authentication and secrecy protocol proposed by Lee, Yang and Hwang [6]. This analysis ascertains a weakness in the employed signature scheme that allows an intruder to forge signatures on arbitrary messages. Two attacks are demonstrated that exploit this weakness and enable an intruder to successfully impersonate mobile users and base stations, respectively. Consequently, this also demonstrates that the protocol fails to achieve its goals of authentication and of privacy of the messages exchanged by the mobile users. Corrections to the flawed protocol are proposed that prevent these weaknesses and the immunity of the amended protocol against the presented attacks is established.

2 The LYH Mobile End-to-End Authentication and Secrecy Protocol

The security protocol proposed by Lee, Yang and Hwang [6] aims to provide secrecy and end-to-end authentication between two mobile users that are connected to different base stations. This protocol is divided into the following phases:

Certification Phase: A trusted certification authority (CA) distributes signed certificates to all protocol participants: Each mobile user and base station will receive only its own certificate. As the certification authority is only involved in the certification phase, it can be considered an offline certification authority.

Authentication Phase: During this phase, the two mobile users attempt to authenticate each other and to establish a secure common session key that is used in the subsequent communication phase.

Communication Phase: The two mobile users use the established session key to communicate securely with each other.

2.1 Certification Phase

Initially, the CA chooses three publicly known parameters (p, q, g): p is a large randomly selected prime number, q is a large prime factor of p-1 and $g = m^{(p-1)/q} \bmod p$, where m is an integer that satisfies 1 < m < p-1 and $m^{(p-1)/q} \bmod p >$ 1. The CA then calculates its own public/private key pair (y_{CA},x_{CA}), where x_{CA} is a randomly chosen private key and y_{CA} is the public key such that $y_{CA} = g^{x_{CA}} \bmod p$. Both x_{CA} and y_{CA} are members of GF(p)*. Then, each principal generates a public/private key pair with the corresponding properties and gives the public key as well as some proof of identity to the CA. The CA then creates the signed certificates (including certificate serial number, validity period, the identity (ID), the corresponding public key etc.) and issues them back to the owning principal.

Validity of CA's Signature. In general, a message signed by the CA consist of the triple (M, s, t), where M is the message itself, s is a random component of the signature and t is the message dependant component of the signature. The value s is calculated by formula (1), where r is a random number selected freshly for each individual signature and h(.) is a 2m-bit iterated hash function that is based on a m-bit block cipher with 2m-bit key. If the underlying block cipher has no weakness, then the hash function can be expected to have ideal computational security against the five attacks: target attack, free-start target attack, collision attack, semi-free-start collision attack and free-start collision attack [9].

$$s = g^r \bmod p \tag{1}$$

The message dependant component *t* of the signature is calculated according to formula (2) using the same value *r* and hash function h(.) as for *s*.

$$t = -s - h(M)x_{CA}{}^{-1} r \bmod q \tag{2}$$

Given a signed message (*M, s, t*), any principal can verify the validity of CA's signature (*s, t*) by establishing that equation (3) holds.

$$y_{CA}^{s+t} s^{h(M)} = 1 \bmod p \tag{3}$$

Assuming that (*s, t*) is a genuine signature, the calculation in (4) is performed to prove validity of the signature.

$$\begin{aligned} & y_{CA}{}^{s+t}(s)^{h(M)} \\ & y_{CA}{}^{s-s-h(M)x_{CA}^{-1}r \bmod q}(s)^{h(M)} \\ & y_{CA}{}^{-h(M)x_{CA}^{-1}r \bmod q}(s)^{h(M)} \\ & (g^{x_{CA}} \bmod p)^{-h(M)x_{CA}^{-1}r \bmod q}(g^r \bmod p)^{h(M)} \\ & (g^{-h(M)r \bmod q} g^{h(M)r}) \bmod p \\ & g^{h(M)r-h(M)r \bmod q} \bmod p \\ & g^0 \bmod p = 1 \bmod p \end{aligned} \tag{4}$$

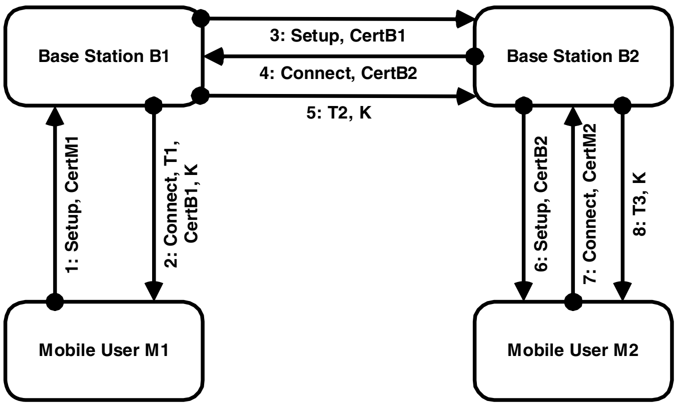


Fig. 1. LYH Protocol with Simplified Messages

2.2 Authentication Phase

In the second phase of the protocol, two mobile users M1 and M2 aim to communicate privately and mutually authenticate each other. Figure 1 outlines the protocol with simplified messages (the detailed messages are presented below in Figures 2, 3 and 4). The protocol can be divided into three parts: The first part comprises of messages 1 and 2 and establishes communication between M1 and B1. The second part consists of messages 3 to 5 and establishes communication between B1 and B2. Finally, the third part includes messages 6, 7 and 8 and establishes communication between B2 and M2.

Communication M1 to B1. The first set of messages establishes communication between the mobile user M1 and its base station B1 using the messages 1 and 2 as detailed in Figure 2.

1. M1 → B1: SETUP, Cert_{M1}
2. B1 → M1: CONNECT, Cert_{B1}, *T1*, $y_{M1}^{-x_{B1}f(T1,DH(B1,M1))}K \bmod p$

Fig. 2. Message exchange between the initiator M1 and the base station B1

M1 sends a SETUP signal and his signed certificate Cert_{M1} to base station B1. B1 verifies the signature using equation (3). If the verification is successful, B1 sends to M1 the CONNECT signal together with B_1's signed certificate, Cert_{B1}, a time stamp *T1* and $y_{M1}^{-x_{B1}f(T1,DH(B1,M1))}K \bmod p$, where f(.) is a secure hash function and DH(B1,M1) is the Diffie-Hellman key [10] between M1 and B1. If the signature on the certificate is validated by equation (3) and the timestamp *T1* is recent, M1 retrieves the session key *K* as demonstrated in equation (5).

$$K = y_{B1}^{x_{M1}f(T1,DH(B1,M1))} y_{M1}^{-x_{B1}f(T1,DH(B1,M1))}K \bmod p \tag{5}$$

Communication B1 to B2. In the second part, communication is established between the base stations B1 and B2 using messages 3, 4 and 5 as detailed in Figure 3. Base station B1 sends a SETUP signal and its certificate Cert_{B1} to B2. Base station B2 verifies the signature of B1's certificate using equation (3).

On successful verification, B2 replies by sending a CONNECT signal and its certificate. If B1 validates B2's certificate, then B1 sends a new timestamp, *T2*, along with $y_{B2}^{-x_{B1}f(T2,DH(B1,B2))}K \bmod p$ to B2.

3. B1 → B2: SETUP, Cert_{B1}
4. B2 → B1: CONNECT, Cert_{B2}
5. B1 → B2: *T2*, $y_{B2}^{-x_{B1}f(T2,DH(B1,B2))}K \bmod p$

Fig. 3. Messages exchanged between the two base stations B1 and B2

Upon receiving message 5 with a valid timestamp, B2 computes the session key *K* using equation (6).

$$K = y_{B1}^{x_{B2} f(T2, DH(B1,B2))} y_{B2}^{-x_{B1} f(T2, DH(B1,B2))} K \bmod p \tag{6}$$

Communication B2 to M1. In the third part, communication is established between the base station B2 and mobile user M2 using messages 6-8 as detailed in Figure 4.

6. B2→ M2: SETUP, $Cert_{B2}$
7. M2→ B2: CONNECT, $Cert_{M2}$
8. B2→ M2: *T3*, $y_{M2}^{-x_{B2} f(T3, DH(B2,M2))} K \bmod p$

Fig. 4. Messages exchanged between the responder M2 and the base station B2

In the third step, B2 sends a SETUP signal to mobile user M_2 along with his certificate $Cert_{B2}$. The mobile user verifies the certificate and sends its certificate along with the CONNECT signal to B2. The base station verifies M2's certificate and sends a new timestamp T_3 along with $y_{M2}^{-x_{B2} f(T3, DH(B2,M2))} K \bmod p$ to M2. The mobile user verifies the validity of the timestamp and computes the session *K* using equation (7).

$$K = y_{B2}^{x_{M2} f(T3, DH(M2,B2))} y_{M2}^{-x_{B2} f(T3, DH(M2,B2))} K \bmod p \tag{7}$$

At this stage, the mobile users M1 and M2 have established a common session key that can be used for private communications.

3 Analyzing the LYH Protocol

The authors of the protocol claim that their protocol correctly achieves its goals of authentication and secrecy [6]. Authentication is ensured by the validity of the certification authority's signature on the public key certificates. To bind a public key to its owner, principals establish whether equation (3) holds, where *M* is replaced by $Cert_X$ as detailed in equation (8). If this is true, then principals will accept that the key contained in the certificate belongs to the principal named in the certificate.

$$y_{CA}^{s+t} s^{h(Cert_x)} = 1 \bmod p \tag{8}$$

Binding principals' IDs to their public keys is crucial for the establishment of the Diffie-Hellman keys, which are used in the distribution of the session key. However, in this paper it will be demonstrated that the employed signature scheme is not secure. It will be shown, that an intruder (also called attacker) can create seemingly correct and valid signatures on any arbitrary message. This is possible without the intruder knowing the Ca's private key x_{CA}. Thus, an intruder can forge the CA's signature on any bogus certificate. Therefore, the security requirements of the protocol are not met.

3.1 Forging Certificates for the LYH Protocol

Consider any arbitrary message *M* and its signature (*s*,*t*) as defined by equations (1) and (2). A principal will consider this signature valid, if it satisfies equation (3). In order to forge CA's signature on message *M*, the intruder needs to establish the pair (*s, t*) such that (*M, s, t*) satisfies equation (3). This can be achieved by selecting s equal to CA's public key y_{CA}. This is equivalent to setting the random value r equal to CA's private key x_{CA} as shown in equation (9) and (10). Note, that it is not required for the attacker to know the value of x_{CA}: To calculate t the attacker simply uses the value 1 for the factor $x_{CA}^{-1}r(=x_{CA}^{-1}x_{CA}=1)$ and it follows that s is equal to CA's public key y_{CA}.

$$s = y_{CA} \bmod p \tag{9}$$
$$t = -y_{CA} - h(M)x_{CA}^{-1}x_{CA} \bmod q = -y_{CA} - h(M) \bmod q \tag{10}$$

A principal that tries to establish the validity of the signature (s, t) on the message M attempts to ascertain if equation (3) holds by performing the calculations shown in (11). As demonstrated, the forged signature satisfies the equation and any principal will accept the forged signature as a legitimate signature of CA. Forging the signature of CA is possible, as the random number r in t appears only as a factor of a term containing x_{CA}^{-1}.

$$\begin{aligned} & y_{CA}^{s+t}s^{h(M)} = 1 \bmod p \\ & y_{CA}^{y_{CA}-y_{CA}-h(M) \bmod q}\, y_{CA}^{h(M)} \bmod p = 1 \bmod p \\ & y_{CA}^{h(M)-h(M) \bmod q} \bmod p = y_{CA}^{0} \bmod p = 1 \bmod p \end{aligned} \tag{11}$$

The intruder can now easily create arbitrary certificates for malicious activities. As trust among principals is established through certificates, an intruder can create certificates to obtain illegitimate privileges. Below, two attacks are presented, in which it is assumed that the intruder has forged a certificates in the presented way to impersonate principals.

3.2 An Attack to Impersonate a Mobile User

The intruder chooses any private key x_I from GF(p)* with its corresponding public key $y_I = g^{x_I} \bmod p$ and creates a mobile-type certificate Cert_I that includes the chosen bogus public key y_I and the identity of the mobile M1. The signature (s_I,t_I) for this bogus certificate is obtained as per equations (9) and (10), i.e. $s = y_{CA} \bmod p$ and $t_I = -y_{CA} - h(Cert_I) \bmod q$. Using this bogus certificate the attacker can initiate a protocol run as M1 to communicate with the mobile user M2 as shown in Figure 5. Neither the mobile user M2 nor any of the base stations B1 or B2 can detect that an illegal certificate is used. Note that the same bogus certificate could be used to take part in the protocol as responder.

In this attack the intruder makes a request to base station B1. The certificate that is presented to the base station appears to be valid. Therefore, B1 is fooled into believing that the intruder is the legitimate mobile user M1. The base station follows the protocol faithfully and generates the session key K, which the intruder I receives in the second message. As the base station B1 used the bogus key in the forged

1. M1(I) → B1: SETUP, Cert_I
2. B1 → M1(I): CONNECT, Cert_{B1}, *T1*, $y_I^{-x_{B1}f(T1,DH(B1,M1))}K \bmod p$
3. B1 → B2: SETUP, Cert_{B1}
4. B2 → B1: CONNECT, Cert_{B2}
5. B1 → B2: T_2, $y_{B2}^{-x_{B1}f(T2,DH(B1,B2))}K \bmod p$
6. B2 → M2: SETUP, Cert_{B2}
7. M2 → B2: CONNECT, Cert_{M2}
8. B2 → M2: *T3*, $y_{M2}^{-x_{B2}f(T3,DH(B2,M2))}K \bmod p$

Fig. 5. Attack to impersonate a mobile user

certificate to protect the session key, the intruder is able to retrieve it. The base station B1 continues the protocol run with B2 and eventually the mobile user M2 receives the session key *K*. M2 believes that *K* is shared with a legitimate user when in fact it is shared with the illegitimate intruder.

3.3 An Attack to Impersonate a Base Station

In another scenario the attacker can forge the signature on a base station-like certificate and compromise the privacy of the communication between two mobile users. The attacker can pose as any of the base stations B1 or B2. When the identity of B1 is assumed, then the intruder is in fact generating the session key *K*. This attack, presented in Figure 6, assumes that the intruder selects a public/private key pair x_I, y_I, creates a base station-like certificate Cert_I and signs it as outlined in (9) and (10). Cert_I contains the selected public key and the identity of B1. Alternatively, the intruder can also masquerade as base station B2. In this case, both B1 and M2 are convinced to accept the intruder as a legitimate base station. As a result, the intruder is able to obtain the session key *K* generated by B1. In both variants of the attack, the intruder possesses the session key *K* and can listen on in the communication between the mobile users M_1 and M_2, whereas they believe that they share a secret session key known only to themselves and legitimate base stations. Consequently, the privacy of the messages exchanged by the mobile users is compromised.

1. M1 → B1(I): SETUP, Cert_{M1}
2. B1(I) → M1: CONNECT, Cert_I, *T1*, $y_{M1}^{-x_I f(T1,DH(B1,M1))}K \bmod p$
3. B1 → B2: SETUP, Cert_I
4. B2 → B1: CONNECT, Cert_{B2}
5. B1 → B2: *T2*, $y_{B2}^{-x_I f(T2,DH(B1,B2))}K \bmod p$
6. B2 → M2: SETUP, Cert_{B2}
7. M2 → B2: CONNECT, Cert_{M2}
8. B2 → M2: *T3*, $y_{M2}^{-x_{B2}f(T3,DH(B2,M2))}K \bmod p$

Fig. 6. Attack to impersonate a base station

4 Fixing the LYH Protocol

The previous section demonstrated that flaws in the signature scheme of the LYH protocol allow an attacker to use algebraic simplifications to impersonate mobile users or base stations. A secure signature scheme is required to avoid the presented attacks.

One ad-hoc solution to avoid these attacks is to discard the signatures that can be generated by the intruder, i.e. to disallow any signature where *s* equals the certification authority's public key y_{CA}. While this prevents the attacks presented in this paper, the signature scheme would still be subject to other weaknesses [11]. Thus, we propose to employ the Elgamal signature scheme [12], which does not allow the algebraic simplifications required for the presented attacks.

4.1 Using the Elgamal Signature Scheme in the LYH Protocol

The Elgamal signature scheme uses the parameters (*p, q, g*) with the same properties as in the LYH scheme: *p* is a large prime number, *q* is a large prime factor of *p*-1 and $g = m^{(p-1)/q} \bmod p$, where *m* is an integer that satisfies 1 < m < p-1 and $m^{(p-1)/q} \bmod p$ > 1. A certification authority CA selects its secret key x_{CA}, the corresponding public key $y_{CA} = g^{x_{CA}} \bmod p$ and generates a fresh random number *r* for each signature. The signature on a message *M* is then given by the pair (*s, t*) where *s* and *t* are as defined in (12) and (13). If either *s* or *t* evaluates to 0 then the signature algorithm is restarted.

$$s = g^r \bmod p \tag{12}$$

$$t = r^{-1}(h(M) - x_{CA}s) \bmod q \tag{13}$$

Principals will accept a signature, if it satisfies equation (14).

$$g^{h(M)} = y_{CA}^s s^t \bmod p \tag{14}$$

The security of this signature scheme lies in the difficulty of solving the discrete logarithm problem and in the difficulty of finding collisions for the employed hash function h(.).

4.2 Resistance of Fixed Protocol to the Presented Impersonation Attacks

In the impersonation attacks presented in section 3 the intruder sets the value of *s* equal to the CA's public key – thus, implicitly, setting the value of r to CA's private key x_{CA} without explicit knowledge of the value of x_{CA}. In the Elgamal scheme, r^{-1} is a factor of the two terms *h(M)* and ($-x_{CA}s$). Thus, if *s* is chosen to be y_{CA} – which implies that *r* is equal to x_{CA} – then the second term reduces to -*s*. However, the attacker still needs to know the explicit value of *r* to calculate the first term $r^{-1}h(M)$. However, this is equivalent to knowing CA's private key x_{CA}, which is a contradiction to the basic assumption that nobody other then CA knows CA's private key. Simply choosing the values according to equations (9) and (10) also fails. The calculations in (15) demonstrate that this will not satisfy equation (14).

$$\begin{aligned} s &= y_{CA} \bmod p \\ t &= -y_{CA} - h(M) \bmod q \\ g^{h(M)} &= y^s s^t \\ g^{h(M)} &= (g^{x_{CA}})^{g^{xCA}} (g^{x_{CA}})^{-g^{xCA} - h(M)} \\ g^{h(M)} &= (g^{x_{CA}})^{-h(M)} \\ g^{h(M)} &\neq g^{-x_{CA} h(M)} \end{aligned} \tag{15}$$

5 Conclusions

This paper analysed the LYH mobile end-to-end authentication and secrecy protocol. The analysis revealed a serious new flaw in the used signature scheme that allows an intruder to use algebraic simplifications to forge signatures. In particular, it was demonstrated how to forge the signature of the certification authority (CA). Further, two attacks were presented where an attacker can impersonate a mobile user and a base station, respectively. In both attacks, neither any honest mobile user nor base station can detect that a forged certificate has been used. In consequence, the LYH protocol fails to achieve is goals of authentication and secrecy.

To avoid the presented attacks, modifications to the LYH protocol were proposed incorporating the use of the Elgamal signature scheme. It was further shown that the use of the Elgamal scheme protects the protocol against the presented attacks.

Acknowledgements. This work was partially funded by Science Foundation Ireland - Research Frontiers Programme SFI 07/RFP/CMSF 631.

References

1. Ventuneac, M., Dojen, R., Coffey, T.: Automated Verification of Wireless Security Protocols using Layered Proving Trees. WSEAS Transactions on Communications 5(2), 252–258 (2006)
2. Dojen, R., Zhang, F., Coffey, T.: On the Formal Verification of a Cluster Based Key Management Protocol for Wireless Sensor Networks. In: 27th IEEE International Performance Computing and Communications Conference – Workshop of Information and Data Assurance, pp. 499–506 (2008)
3. Dojen, R., Lasc, I., Coffey, T.: Establishing and Fixing a Freshness Flaw in a Key-Distribution and Authentication Protocol. In: IEEE International Conference on Intelligent Computer Communication and Processing, pp. 185–192 (2008)
4. Hwang, R.J., Su, F.F.: A new efficient authentication protocol for Mobile networks. Computer Standards & Interfaces 28(2), 241–252 (2005)
5. Chien, H., Jan, J.: A hybrid authentication protocol for large mobile network. Journal of System Software 67(2), 123–130 (2003)
6. Lee, C.C., Yang, C.C., Hwang, M.S.: A new privacy and authentication protocol for end-to-end mobile users. International Journal of Communication Systems 16(9), 799–808 (2003)

7. Chang, C., Chen, K., Hwang, M.: End-to-End Security Protocol for Mobile Communications with End-User Identification/Authentication. Wireless Personal Communications: An International Journal 28(2), 95–106 (2004)
8. Park, M., Okazaki, N., Baba, Y.: A New User Authentication Protocol for Mobile Terminals in Wireless Network. In: 7th International Conference on Mobile Data Management (MDM 2006), p. 94 (2006)
9. Yi, X., Lam, K.Y.: Hash function based on block cipher. IEE Electronics Letters 33(23), 1938–1940 (1997)
10. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory, IT 22(6), 644–654 (1976)
11. Chang, C.C., Lee, J.S.: Improvement on an Optimized Protocol for Mobile Network Authentication and Security. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3802, pp. 538–541. Springer, Heidelberg (2005)
12. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

Author Index