John A. Clark
Richard F. Paige
Fiona A.C. Polack
Phillip J. Brooke (Eds.)

LNCS 3934

Security in Pervasive Computing

Third International Conference, SPC 2006
York, UK, April 2006
Proceedings

Springer

John A. Clark   Richard F. Paige
Fiona A.C. Polack   Phillip J. Brooke (Eds.)

Security in Pervasive Computing

Third International Conference, SPC 2006
York, UK, April 18-21, 2006
Proceedings

Springer

Volume Editors

John A. Clark
Richard F. Paige
Fiona A.C. Polack
University of York
Department of Computer Science
Heslington, York, YO10 5DD, UK
E-mail: {jac,paige,fiona}@cs.york.ac.uk

Phillip J. Brooke
University of Teesside
School of Computing
Middlesbrough, TS1 3BA, UK
E-mail: p.j.brooke@tees.ac.uk

Library of Congress Control Number: 2006923045

CR Subject Classification (1998): C.2, D.2, D.4.6, H.5, K.4.1, K.4.4, K.6.5, H.4

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-540-33376-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-33376-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11734666 06/3142 5 4 3 2 1 0

Preface

This volume contains the papers presented at the Third International Conference on Security in Pervasive Computing (SPC 2006), held April 19–21, 2006 in York, UK. The conference focused on methods, tools, principles, and practices for assessing and achieving security in a pervasive environment. New security concepts were discussed, in domains and applications such as handheld devices, mobile phones, smartcards, RFID chips, and smart labels, as well as new, emerging technological spaces. The conference also presented work on fundamental themes such as risk identification and mitigation, security policies for pervasive environments, privacy measures (especially cryptographic protocols), and mobility and location-aware services. Submissions included work on biometrics, ambient intelligence, Web services, security requirements, and many other topics.

We received 56 submissions, and accepted 16 full papers for presentation. Each submission was reviewed by the international Programme Committee. We are grateful to the Programme Committee members, and the additional reviewers, for their timely completion of the reviewing process, and for the quality and detail of their reviews and discussion.

Our thanks go to all members of the Programme Committee for their efforts; the additional reviewers; the authors, for submitting their papers; the keynote speaker, Frank Stajano; the invited speaker, Howard Chivers; and the Department of Computer Science, University of York, for supporting the event.

April 2006

John A. Clark (Program Chair)
Richard F. Paige
Fiona A.C. Polack
Phillip J. Brooke

Organization

SPC 2006 was organized by the Department of Computer Science, University of York.

Executive Committee

Program Chair	John A. Clark (Univ. of York, UK)
Organizing Co-chairs	Richard F. Paige and Fiona A.C. Polack (Univ. of York, UK)
Publicity Chair	Phillip J. Brooke (Univ. of Teesside, UK)

Programme Committee

Anos Anastassiadis	Cyveillance, USA
N. Asokan	Nokia, Finland
Phil Brooke	Univ. of Teesside, UK
Howard Chivers	Cranfield University, UK
Stephen J. Crane	HP, UK
Sadie Creese	QinetiQ, UK
Michael Goldsmith	Formal Systems Europe, UK
Stefanos Gritzalis	Univ. of the Aegean, Greece
Jochen Haller	SAP, Germany
Dieter Hutter	DFKI, Germany
Paul Karger	IBM, USA
Dennis Kuegler	BSI, Germany
Marc Langheinrich	ETH Zurich, Switzerland
Cetin Kaya Koc	Oregon State, USA
Cathy Meadows	NRL, USA
Takashi Moriyasu	National Information Security Center, Japan
Guenter Mueller	Univ. of Freiburg, Germany
Richard Paige	Univ. of York, UK
Panos Papadimitratos	Virginia Tech, USA
Fiona Polack	Univ. of York, UK
Yves Roudier	Eurecom, France
Peter Ryan	Univ. of Newcastle, UK
Andrei Serjantov	Free Haven Project, UK
Werner Stephan	DFKI, Germany
Markus Ullman	BSI, Germany
Irfan Zakuiddin	QinetiQ, UK

Additional Referees

Table of Contents

Trust Without Identification

Howard Chivers

Department of Information Systems, Cranfield University, DCMT Shrivenham, Swindon SN6 8LA, UK
hrchivers@iee.org

Abstract. This extended abstract describes an alternative to trusting individual nodes in pervasive systems, which is to exploit the diversity of nodes in such systems to build application ensembles that are collectively trustworthy. These configurations are resilient to high levels of attack, and are not dependent on large pre-distribution key-spaces.

1 Background

Trust is a measure of belief in expected behaviour, in particular the likelihood of a particular outcome of a transaction; inevitably such estimates underpin security decision-making. Trust reputation systems estimate the likely behaviour of a node from the history of its interactions, which include recommendations obtained from other nodes. However, the nodes in such systems must be individually identifiable, otherwise they are vulnerable to an attacker who simply creates multiple electronic persona: the Sybil attack [1]. Such attacks can exploit start-up credits in reputation schemes, or fake many low-value recommendations to build an undeserved reputation.

Establishing a reliable identity in a pervasive system is problematic because nodes may have limited long-term storage and intermittent connectivity. Conventional public key systems require certificate validation and revocation, which may be difficult to achieve. An alternative is to extend key pre-distribution schemes to support identity [2]; a node can be identified by a number of keyspaces, which are probed as other network nodes establish connections.

The identity problem occurs because of the assumption that trust is a property of an individual node; however, in pervasive systems this may be questionable. Pervasive applications exploit the redundancy provided by a large number of nodes to achieve an adequate level of robustness, reliability, and performance. An important question is the extent that this approach can also be used to support security.

The real objective is to trust the outcome of an application, and there are many cases where appropriate outcomes can be achieved even if some of the contributions are in doubt; examples range from simple voting schemes to the sophisticated signal processing of sensor information. Of course, voting would be vulnerable to Sybil, but there are ways of improving the situation without resorting to individual identities.

2 Configuration Trust

A possible approach is to simply ensure that the nodes in an application are different; such an application may include malicious nodes, but up to a threshold they will be unable to overwhelm the application and corrupt its results. This can be achieved by pre-distribution of authentication tokens or *diversity keys*; the application configuration is assembled using nodes that hold different keys.

These ideas can be extended further; in the process of configuring an application, nodes with identical diversity keys may be encountered. The nodes may be legitimate; alternatively, the particular key may be over-represented, indicating an attacker who is replicating nodes to improve their likelihood of use. The more aggressive the attack, the easier it can be rejected. In contrast, conventional trust recommendation schemes are unlikely to converge given a high proportion of malicious nodes.

Related security problems can also be addressed by managing trust at the level of the application configuration, rather than the individual. Location verification can be used to defend against Sybil attacks, but accurately locating individual nodes is an open research question [2]. In many applications, it may be sufficient to ensure that nodes are in a consistent location, rather than measure their actual position. For example, batch identifiers could be implanted in sensors nodes; nodes from the same batch are distributed in a similar way and are likely to be co-located.

These security concepts have been applied to the concrete example of a sensor network [3]. A significant result is that the security of an application is not strongly related to the size of the diversity keyspace. A diversity keyspace equal to the number of network nodes would amount to individual identities, so this result confirms that identity is not an essential precursor to trust in an application.

3 Conclusion

An alternative to trusting individual nodes in a pervasive system, is to focus on the need to trust the outcome of an application, by exploiting diversity. This viewpoint suggests alternative trust protocols that reject very high levels of attack, and are not dependent on large pre-distribution key-spaces. The fact that such protocols can be designed suggests that there is scope for further work in this field, and that it may not be necessary to be able to prove the identity of every node in a pervasive system.

References

1. Douceur, J. R. The Sybil Attack, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, Cambridge, MA (USA). (*Lecture Notes in Computer Science, vol. 2429/2002*). Springer-Verlag, 2002; 251-260.
2. Newsome, J., Shi, E., Song, D., and Perrig, A. The Sybil Attack in Sensor Networks: Analysis & Defenses, *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN'04)*, Berkley California. ACM Press, 2004; 259-268.
3. Chivers, H. and Clark, J. A., Smart dust, friend or foe?—Replacing identity with configuration trust. *Computer Networks Special Issue: Military Communications Systems and Technologies*, 2004. **46**(5). 723-740.

Constant-Round Password-Based Group Key Generation for Multi-layer Ad-Hoc Networks[⋆]

Jin Wook Byun[1], Su-Mi Lee[1], Dong Hoon Lee[1], and Dowon Hong[2]

[1] Center for Information Security Technologies (CIST),
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
{byunstar, donghlee, smlee}@korea.ac.kr
[2] Electronics and Telecommunications Research Institute (ETRI),
Gajeong-dong, Yuseong-gu, Daejeon, Korea
dwhong@etri.re.kr

Abstract. In this paper, we consider a multi-layer mobile ad-hoc network (MANET) composed of several kinds of networking units (such as ground soldiers, tanks, and unmanned aerial vehicles) with heterogeneous resources to communicate and compute. In this multi-layer MANET, we first propose a password-based authenticated group key exchange scheme with members' different passwords. The proposed scheme only requires constant-round to generate a group session key under the dynamic scenario, hence it is scalable, i.e., the overhead of key generation is independent of the size of a total group. We support the proposed scheme with formal security proof. Namely, our proposed scheme is the first constant-round password-based group key exchange with different passwords for the dynamic setting of MANET.

Keywords: Password authentication, key agreement, authenticated key exchange, heterogeneous, pervasive computing, multi-layer ad-hoc network.

1 Introduction

A mobile ad-hoc network (MANET) is a wireless network composed of mobile nodes that require little or no fixed infrastructure to communicate, and it has dynamic property itself because any mobile node may join and also leave the network at any given time. Thus, to protect communication between mobile nodes, it is desirable to use efficient and scalable cryptographic solutions with dynamic configuration. To communicate securely over an insecure wireless MANET it is essential that secret keys (encryption and decryption keys) are exchanged securely. A password-based authenticated group key exchange protocol allows mobile nodes holding passwords to agree on a common secret key over an insecure ad-hoc network in a secure and authenticated manner.

[⋆] This work was supported by grant No. R01-2004-000-10704-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

1.1 Related Works

In this paper, we first consider the problem of a password-based group *Diffie-Hellman* key exchange in a dynamic scenario using *different* passwords. Most password-based authenticated key exchange schemes in the literature have focused on an authenticated key exchange using a *shared* password between clients or between a client and a server [1, 4, 5, 8, 20, 21, 24, 29, 15, 19]. However, the setting such that all clients have a same password is not practical since a password is not a common secret but a secret depending on an individual. For example, in mission-critical MANET such as emergency rescue and military operations, the setting in which group mobile nodes have different passwords is more suitable.

In recent years, a lot of password-based key exchange using different passwords have been presented. Byun *et al.* first proposed a secure *Client-to-Client Password-Authenticated Key Agreement* (C2C-PAKA) in the cross-realm setting where two clients were in two different realms and hence there existed two servers involved [10]. They have heuristically proved that the schemes were secure against all attacks considered. Unfortunately, the scheme was found to be flawed. Chen first pointed out that in the scheme with the cross-realm setting one malicious server can mount a dictionary attack to obtain the password of client who belongs to the other realm [13]. In [9], Wang *et al.* showed three dictionary attacks on the same protocol, and Kim *et al.* pointed out that the protocol was susceptible to Dening-Sacco attack in [23]. Kim *et al.* also proposed a improved C2C-PAKA protocol. However, very recently, Phan and Goi suggested two unknown key share attacks on the improved C2C-PAKA protocol in [25]. Several countermeasures to protect the attacks on C2C-PAKA protocol have been presented in [13, 9, 23, 25], but without any formal treatment. Very recently, Byun *et al.* efficiently and securely modified the original C2C-PAKA protocol with formal security model and proof, and presented EC2C-PAKA protocol [12].

In the group setting, Byun and Lee first suggested a password-based group key exchange protocols using group members' different passwords [9], but the schemes also had security holes such that a malicious insider can get information of other valid users' passwords [28]. Very recently, Byun *et al.* revised the schemes of [9], and presented password-based group key exchange schemes secure against insider guessing attacks [11]. However, these protocols do not consider dynamic scenario in which a group member is not known in advance but any member may join and also leave the group at any time, which is one of the most important properties in MANET.

1.2 Our Contributions

In this paper, we study a password-based group key exchange for the dynamic MANET. In particular, we focus on a heterogeneous multi-layer MANET with mobile backbone and unmanned aerial vehicles, which have been studied in [16, 17, 18, 26]. Consequently, we first design a constant-round password-based group key exchange protocol (namely, N-party EKE-D) using different passwords for a multi-layer MANET setting. As illustrated in Figure 1, the multi-layer

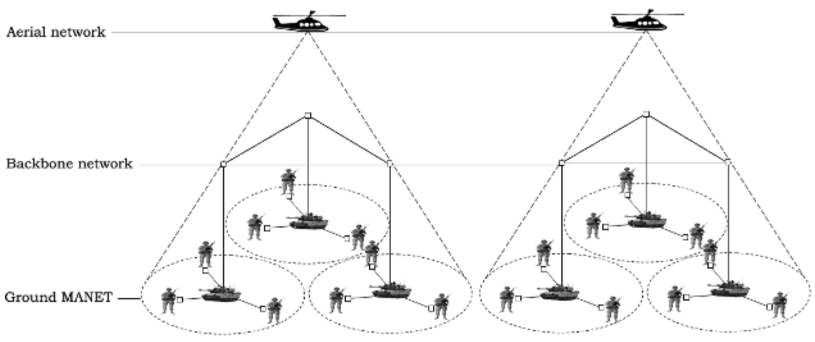


Fig. 1. The framework of Multi-layer Mobile Ad-hoc Network

MANET is composed of three kinds of networking units with heterogeneous communication capabilities and computation powers: the regular ground mobile nodes, the ground mobile backbone nodes, and the unmanned aerial vehicle nodes. Their capabilities and characteristics are summarized as follows [16, 17, 18, 26].

- Ground MANET (first layer) : This network includes regular ground mobile (GN) nodes and a ground mobile backbone (MBN) node. For instance, GN nodes can be soldiers equipped with limited communication and computation devices. They have the constrained transmission capabilities and use the short range channel access.
- Ground mobile backbone network (second layer) : This network includes MBN nodes which are special fighting units like trucks and tanks. These MBN nodes may carry a lot more equipment than individual soldiers.
- Unmanned aerial vehicles (third layer) : The unmanned aerial vehicles (UAV) maintain a station at an altitude of 10 to 20 thousands feets by flying in a circle. With the help of phased array antennas, it can provide the shared beam to the ground to keep line-of-sight connectivity for one area of operation down below.

The multi-layer MANET assumes that the second and third layers have well established infrastructures equipped with more communication and computation powers than the ones of the first layer. That is, highly cost cryptographic solutions can be used to protect communication between MBN nodes (tanks and trucks) and UAV nodes (such as airlines). Thus, we may employ wireless PKI-based group key exchange and tree-based group key exchange protocols [6, 22] as a security module for the second and third layer networks. However, in the first ground MANET, there are various physical attacks and bombs with dynamic changing network configuration, hence quick and secure cryptographic solutions should be applied to protect communication of GN nodes such as soldiers. Since our scheme only uses human memorable passwords to generate a group session key without any public key infrastructure (PKI) requiring tedious and expensive certificate management, it is well suitable for making a secure channel between soldiers in the first layer ground MANET. In addition, our proposed N-party EKE-D protocol requires only a constant number of rounds to establish a group session key. Accurately, one round is demanded by nodes, and two rounds are demanded by a central MBN node.[1] Furthermore only 2 modular exponentiations are required by each GN node. Our proposed N-party EKE-D protocol is the first constant-round and provably secure scheme in the dynamic scenario. We show that our proposed scheme is secure under the assumption that computational *Diffie-Hellman* problem is intractable.

2 Security Model and Definition

In this section we briefly review communication model and security definition for designing a secure password-based group key exchange protocol. Our model do not include the general security model for the multi-layer MANET, but just only security model for the ground MANET. Our communication model are based on the works of [1, 5].

2.1 Communication Model

Participants. We have two types of protocol participants, *GNs* and *MBNs*. Let $ID = GNs \cup MBNs$ be a non-empty set of protocol participants. We assume that *MBNs* consists of a single central node MBN, and $GNs = \{\mathsf{GN}_1, ..., \mathsf{GN}_n\}$ consists of identities of n GN nodes. Each node $\mathsf{GN}_i \in GNs$ has a secret password pw_i, and central MBN keeps password verifiers in its database. A node $\mathsf{GN}_i \in GNs$ may execute a key exchange protocol multiple times with different partners, and we denote the t-th instance of the protocol executed by entity GN_i (MBN) as oracle GN_i^t (MBN_i^t, respectively).

Algorithm. An N-party EKE-D protocol P requires the following four algorithms.

- **Password Generation Algorithm** $\mathsf{PGA}(1^k)$: Given an input of 1^k, where k is a security parameter, and then provides each node $\mathsf{GN}_i \in GNs$ with password pw_i.
- **Setup Algorithm** $\mathsf{Setup}(\mathcal{C})$: Takes input as a set of $\mathcal{C}$ and starts the protocol P. A new set $\mathcal{I}$ is created and set by $\mathcal{I} = \mathcal{C}$.
- **Join Algorithm** $\mathsf{Join}(\mathcal{I}, \mathcal{J})$: $\mathcal{J}$ is a set of newly joining group members. Takes inputs as sets $\mathcal{I}$ and $\mathcal{J}$, updates $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$. Output is a new group session key shared between nodes of $\mathcal{I}$ including newly joining group members.

[1] We assume that ground MANET is the multicast network, hence any node can send messages to multiple recipients only in one round. One round includes all the messages that can be sent in parallel during the protocol.

- **Remove Algorithm** $\mathsf{Remove}(\mathcal{I}, \mathcal{R})$: $\mathcal{R}$ is a set of leaving group members. Takes inputs sets $\mathcal{I}$ and $\mathcal{R}$, updates $\mathcal{I} = \mathcal{I} \backslash \mathcal{R}$. Output is a new group session key shared between nodes of $\mathcal{I}$ excluding the leaving group members.

We allow the adversary to potentially control all communication in the network via access to a set of oracles as defined below. We consider an *experiment* in which the adversary asks queries to oracles, and the oracles answer back to the adversary. Oracle queries model attacks which the adversary may use in the real system. We consider the following types of queries in this paper.

- A query $\mathsf{Send}(\mathsf{GN}_i^t, M)$ is used to send a message M to instance GN_i^t. When GN_i^t receives M, it responds according to the key-exchange protocol. The adversary may use this query to perform *active* attacks by modifying and inserting the messages of the key-exchange protocol. Hence, impersonation attacks and man-in-the-middle attacks are possible using this query.
- A query $\mathsf{Execute}(\mathit{GNs})$ represents passive eavesdropping of the adversary on an execution of the protocol between honest nodes in *GNs*. Namely, all nodes in *GNs* execute the protocol without any interference from the adversary, and the adversary is given the resulting transcript of the execution.
- A query $\mathsf{Reveal}(\mathsf{GN}_i^t)$ models the *known key* attacks in the real system. The adversary is given the session key of the specified instance GN_i^t.
- A query $\mathsf{Corrupt}(\mathsf{GN}_i^t)$ models exposure of the long-term password of GN_i. The adversary is assumed to be able to obtain long-term passwords of nodes, but cannot control the behavior of these nodes directly (of course, once the adversary has asked a query $\mathsf{Corrupt}(\mathsf{GN}_i)$, the adversary may impersonate GN_i in subsequent Send queries).
- A query $\mathsf{Test}(\mathsf{GN}_i^t)$ is used to define the advantage of an adversary. If GN_i^t is a fresh oracle (defined in Section 2.3), then the oracle GN_i^t flips a coin b. If b is 1, then a session key is returned. Otherwise, a string randomly drawn from a session key distribution is returned. The adversary is allowed to make a single Test query, at any time during the experiment.

2.2 Security Definition and Computational Assumption

Definition 2.1 [CDH : Computational Diffie-Hellman]. Let $\mathcal{A}_\Delta$ be **CDH** adversary running in polynomial time T_Δ. The **CDH** problem is defined as follows: Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q. Given a group $\mathbb{G}$, a generator g of $\mathbb{G}$, and two elements g^a and $g^b \in \mathbb{G}$, where a and b are unknown, compute g^{ab}. An algorithm $\mathcal{A}_\Delta$ running in time T_Δ is said to solve the **CDH** problem with the an advantage $\mathbf{Adv}_{\mathbb{G}}^{cdh}(T_\Delta)$ of ϵ if:

$$|\Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b) = g^{ab}]| \geq \epsilon.$$

We say the **CDH** assumption holds in $\mathbb{G}$ if no probabilistic polynomial time algorithm $\mathcal{A}_\Delta$ can solve the **CDH** problem with non-negligible advantage.

Definition 2.2 [Collision-Resistant Hash Function]. Let $\mathbb{F} = \{f_k\}_{k \in \{0,1\}^l}$ be a polynomial time computable family of functions for every $k \in \{0,1\}^l$.

A function $f_k : \{0,1\}^* \rightarrow \{0,1\}^l$ is a collision-resistant hash function if the followings are satisfied.

- For $x \in \{0,1\}^*$, it is easy to find a value $y \in \{0,1\}^l$ such that $f_k(x) = y$.
- For every polynomial-time adversary $\mathcal{A}$, the following probability is negligible.[2]

$$\Pr[\mathcal{A}(k) = (x,y) | f_k(x) = f_k(y) \wedge x \neq y] \leq \epsilon(k)$$

Security Definition. We focus on dynamic scenario where any group member may join and also leave the group. Thus, the goal of our protocols is for current participants to share a common session key sk which is known to nobody (including revoked group members) but participants under passive or active adversaries. In the security definition, we consider the following properties. The output of real session key must be indistinguishable from a random key by the adversary. However we note that if an adversary initiates m ($\leq |\mathcal{D}|$, which is a size of dictionary) instances of a password-based authenticated key exchange protocol, and guesses an appropriate password in each initiation, then it will succeed in guessing the password with probability $m/|\mathcal{D}|$. So the given N-party EKE-D (NEKED) protocol is considered secure if the active adversary can not do significantly better than this trial bound. In addition, in the dynamic scenario, we consider additional security requirements (such as forward secrecy and backward secrecy) against the revoked GN nodes and newly joined GN nodes. With this knowledge we define the security of NEKED as follows.

Definition 2.3 [NEKED Security]. Consider the following experiment. Firstly, a password is assigned to each node by running the password generation algorithm $\mathsf{PGA}(1^k)$, and the password is registered in the server MBN by running the registration algorithm $\mathcal{R}$. Then an adversary $\mathcal{A}$ is run. It will interact with finite oracles by asking queries defined above during the experiment. The adversary $\mathcal{A}$ can ask a Test query to a fresh oracle only once at any time. When $\mathcal{A}$ asks a Test query, then experiment flips a coin for bit b. If it lands $b = 1$, a real session key is returned to the adversary. Otherwise, a random key is returned to the adversary. After $\mathcal{A}$ is given a random or a real key, $\mathcal{A}$ may ask other queries continuously and perform an on-line password guessing attacks adaptively. Eventually $\mathcal{A}$ guesses b, outputs the guessed bit b' and terminates the experiment. Let $\mathbf{Succ}_{\mathcal{A}}^{neke}$ be the event that $b' = b$. The session key advantage of $\mathcal{A}$ in attacking protocol P is defined as

$$\mathbf{Adv}_P^{neked}(\mathcal{A}) = 2Pr[\mathbf{Succ}_{\mathcal{A}}^{neked}] - 1.$$

To define completeness of the protocol, we use the concept of a *benign adversary* which was firstly defined in [3]. A benign adversary only relays the transmitted messages between participants faithfully. A given N-party EKE-D protocol P *is secure if* the following conditions are satisfied:

[2] A function $\nu(k)$ is a negligible function if for every constant $c \geq 0$ there exists an integer k_c such that $\nu(k) < k^{-c}$ for all $k \geq k_c$.

(1) **Completeness:** In the presence of a benign adversary all partnering accept the same session key.
(2) **Indistinguishability:** Formally, for all probabilistic polynomial time adversary $\mathcal{A}$, every finite dictionary $\mathcal{D}$, and for all $m \leq |\mathcal{D}|$, the session key advantage of $\mathcal{A}$ should be bounded as follows.

$$\mathbf{Adv}_P^{neked}(\mathcal{A}) \leq \frac{m}{|\mathcal{D}|} + \varepsilon(k).$$

where $\varepsilon(k)$ is a negligible function and k is a security parameter.
(3) **Forward Secrecy:** Many soldiers (GN nodes) in the ground MANET may be killed on the battlefield by various bombs and physical attacks, and a fresh supply of soldiers may be frequently required. At that time, newly joining soldiers should not decrypt secret data sent before they join the group. That is, newly joined GN nodes cannot gain access to past group session keys.
(4) **Backward Secrecy:** After GN nodes left their current group, the GN nodes should not be able to gain access to future group session keys. To provide the forward and backward secrecy, a newly group session key should be generated by Join and Remove algorithms.

3 N-Party EKE-D Protocol

3.1 Setup($\mathcal{C}$)

For an initial group $\mathcal{C} = \{\mathsf{GN}_1, ..., \mathsf{GN}_n\}$, the Setup algorithm consists of four sub-algorithms $\mathsf{Setup}^1(\mathcal{C})$, $\mathsf{Setup}^2(\mathcal{C})$, $\mathsf{SerCom}(\mathcal{C})$, and $\mathsf{NodeCom}(\mathcal{C})$ as described in Figure 2. $\mathcal{H}_i$ is an ideal hash function such that $\mathcal{H}_i : \{0,1\}^* \rightarrow \{0,1\}^l$ for $1 \leq i \leq 3$.

- **$\mathsf{Setup}^1(\mathcal{C})$** : In the first round, the single server MBN sends $\mathcal{E}_{pw_i}(g^{s_i})$ to GN_i nodes concurrently for $1 \leq i \leq n$. Simultaneously, each node GN_i also sends $\mathcal{E}_{pw_i}(g^{x_i})$ to the single-server MBN concurrently in the first round. After finishing the first round, MBN and GN_i share an ephemeral *Diffie-Hellman* key, $sk_i = \mathcal{H}_1(sid'||g^{x_i s_i})$ where $sid' = \mathcal{E}_{pw_1}(g^{x_1})||\mathcal{E}_{pw_2}(g^{x_2})||...||\mathcal{E}_{pw_n}(g^{x_n})$.
- **$\mathsf{SerCom}(\mathcal{C})$** : For all nodes $\mathsf{GN}_i \in \mathcal{C}$ $(1 \leq i \leq n)$, MBN secretly computes $\overline{sk_i} = F(sk_i)$ and saves a set $\mathtt{List}_{sc} = \{\overline{sk_1}, .., \overline{sk_n}\}$. F is a collision resistant hash function from $\{0,1\}^*$ to $\{0,1\}^l$.
- **$\mathsf{Setup}^2(\mathcal{C})$** : In the second round, MBN selects a random value N from Z_q^* and hides it by exclusive-or operation with the ephemeral key $\overline{sk_i}$ from $\mathtt{List}_{sc} = \{\overline{sk_1}, .., \overline{sk_n}\}$. MBN broadcasts $N \oplus \overline{sk_i}$ and authenticator $\mathcal{H}_2(\overline{sk_i}||\mathsf{MBN})$ to GN_i for $1 \leq i \leq n$. Concurrently, GN_i nodes broadcast authenticators $\mathcal{H}_2(\overline{sk_i}||\mathsf{GN}_i)$ for $\overline{sk_i}$, respectively. MBN and GN_i checks that its authenticator is valid by using $\overline{sk_i}$. After finishing the second round, all nodes can get a random secret N using its $\overline{sk_i}$, and generate a group session key, $sk = \mathcal{H}_3(SIDS||N)$ where $SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||...||sk_n \oplus N$.
- **$\mathsf{NodeCom}(\mathcal{C})$** : For $1 \leq i \leq n$, all nodes $\mathsf{GN}_i \in \mathcal{C}$ compute $\overline{sk_i} = F(sk_i)$ and save it in the secure storage of nodes. Lastly, a new set $\mathcal{I}$ is created, and set by $\mathcal{I} = \mathcal{C}$.

To add the mutual authentication (key confirmation) to N-party EKE-D protocol, we can use the additional *authenticator* $\mathcal{H}_4(sk||i)$ described in [6].

	MBN	GN_1	GN_2	...	GN_n
$\mathsf{Setup}^1(\mathcal{J})$	$s_i \leftarrow [1, q-1]$ $\mathcal{E}_{pw_i}(g^{s_i})$	$x_1 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_1}(g^{x_1})$	$x_2 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_2}(g^{x_2})$	... ...	$x_n \leftarrow [1, q-1]$ $\mathcal{E}_{pw_n}(g^{x_n})$
$\mathsf{SerCom}(\mathcal{J})$	$\mathtt{List}_{sc} = \{\overline{sk}_1, .., \overline{sk}_n\}$				
$\mathsf{Setup}^2(\mathcal{J})$	$\boxed{\mathcal{H}_2(\overline{sk}_i\|\|S)}$	$\boxed{\mathcal{H}_2(\overline{sk}_1\|\|\mathsf{GN}_1)}$	$\boxed{\mathcal{H}_2(\overline{sk}_2\|\|\mathsf{GN}_2)}$	...	$\boxed{\mathcal{H}_2(\overline{sk}_n\|\|C_n)}$
	$N \leftarrow [1, q-1]$ $sk_1 \oplus N\|\|...\|\|sk_n \oplus N$	sk	sk	...	sk
$\mathsf{NodeCom}(\mathcal{J})$		$\overline{sk}_1$	$\overline{sk}_2$	...	$\overline{sk}_n$

Fig. 2. SetUp Algorithm of N-party EKE-D

Notation. $sk_i (= \mathcal{H}_1(sid'||g^{x_i s_i}))$ is an ephemeral key generated between MBN and node GN_i in the first round, where $sid' = \mathcal{E}_{pw_1}(g^{x_1})||\mathcal{E}_{pw_2}(g^{x_2})||...||\mathcal{E}_{pw_n}(g^{x_n})$. A common group session key among nodes is $sk = \mathcal{H}_2(SIDS||N)$, where $SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||...||sk_n \oplus N$ and N is a random value chosen from $[1, q-1]$.

3.2 Join($\mathcal{I}, \mathcal{J}$)

By bombs and various physical attacks, a central MBN node can be destroyed. In this case, GN nodes of the destroyed MBN should quickly join a new MBN node by performing Join algorithm. As illustrated in Figure 3, the main idea to deal with newly joined GN nodes is that MBN node firstly generates intermediate keys with newly joined GN nodes by executing Setup^1 algorithm, and then all GN nodes generate a group session key with MBN node by executing Setup^2 algorithm. Without loss of generality, we assume that $\mathcal{I} = \{\mathsf{GN}_1, .., \mathsf{GN}_n\}$ and $\mathcal{J} = \{\mathsf{GN}_{n+1}, ..., \mathsf{GN}_{n+j}\}$. The Join algorithm first takes inputs as a current set $\mathcal{I}$ and a joining set $\mathcal{J}$, and then updates $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$.

- In the first round, Join algorithm executes $\mathsf{Setup}^1(\mathcal{J})$ and $\mathsf{SerCom}(\mathcal{J})$. After executing $\mathsf{SerCom}(\mathcal{J})$, MBN gets a new set $\mathtt{List}'_{sc} = \{sk_{n+1}, .., sk_{n+j}\}$ which consists of intermediate *Diffie-Hellman* keys $\overline{sk}_j$ for $j \in \mathcal{J}$. MBN updates $\mathtt{List}_{sc}$ by $\mathtt{List}_{sc} \cup \mathtt{List}'_{sc}$.
- In the second round, Join algorithm executes $\mathsf{Setup}^2(\mathcal{I})$ algorithm with an input of a set $\mathcal{I}(= \mathcal{I} \cup \mathcal{J})$. More precisely, for $1 \leq i \leq n+j$, MBN selects a random value N from Z_q^* and computes $N \oplus \overline{sk}_i$ with $\mathtt{List}_{sc} = \{\overline{sk}_1, .., \overline{sk}_{n+j}\}$. MBN broadcasts $N \oplus \overline{sk}_i$ and authenticator $\mathcal{H}_2(\overline{sk}_i||\mathsf{MBN})$ to GN_i. Each

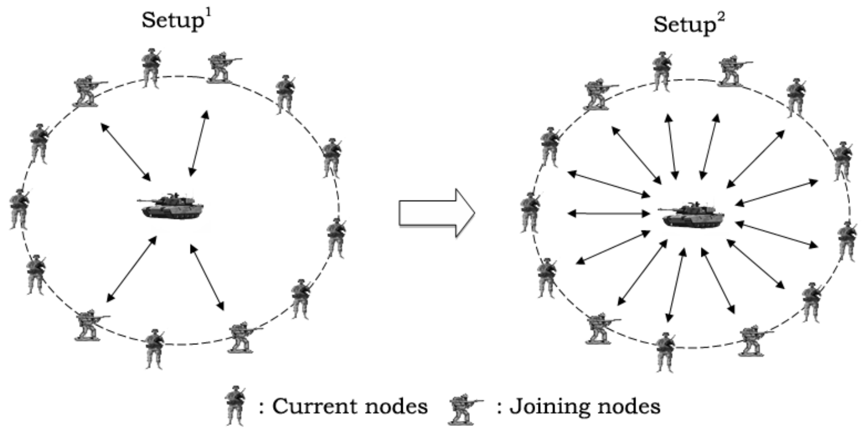


Fig. 3. The framework of Join Algorithm

node $\mathsf{GN}_i \in \mathcal{I}$ also broadcasts authenticators $\mathcal{H}_2(\overline{sk_i}||\mathsf{GN}_i)$. MBN and GN_i check that its authenticator is valid by using $\overline{sk_i}$. Consequently, all nodes can get a random secret N using its $\overline{sk_i}$, and generate a group session key, $sk = \mathcal{H}_3(SIDS||N)$ where $SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||...||sk_{n+j} \oplus N$ and $sid' = \mathcal{E}_{pw_1}(g^{x_1})||\mathcal{E}_{pw_2}(g^{x_2})||...||\mathcal{E}_{pw_{n+j}}(g^{x_{n+j}})$.

For $\mathcal{I} = \{\mathsf{GN}_1, \mathsf{GN}_2\}$, $\mathcal{J} = \{\mathsf{GN}_3\}$ we illustrate $\mathsf{Join}(\mathcal{I}, \mathcal{J})$ algorithm in Figure 4.

	MBN	GN_1	GN_2	$\boxed{\mathsf{GN}_3}$
	$\mathtt{List}_{sc} = \{\overline{sk_1}, \overline{sk_2}\}$	$\overline{sk_1}$	$\overline{sk_2}$	
Setup$^1(\mathcal{J})$	$s_3, s_4 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_i}(g^{s_i})$			$x_3 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_3}(g^{x_3})$
SerCom$(\mathcal{J})$	$\mathtt{List}_{sc} = \{\overline{sk_1}, \overline{sk_2}, \overline{sk_3}, \overline{sk_4}\}$			
Setup$^2(\mathcal{J})$	$\boxed{\mathcal{H}_2(\overline{sk_i}\|\|S)}$	$\boxed{\mathcal{H}_2(\overline{sk_1}\|\|\mathsf{GN}_1)}$	$\boxed{\mathcal{H}_2(\overline{sk_2}\|\|\mathsf{GN}_2)}$	$\boxed{\mathcal{H}_2(\overline{sk_3}\|\|\mathsf{GN}_3)}$
	$N \leftarrow [1, q-1]$ $sk_1 \oplus N\|\|...\|\|sk_4 \oplus N$	sk	sk	sk
NodeCom$(\mathcal{J})$		$\overline{sk_1}$	$\overline{sk_2}$	$\overline{sk_3}$

Fig. 4. Join Algorithm of N-party EKE-D for $\mathcal{I} = \{\mathsf{GN}_1, \mathsf{GN}_2\}$, $\mathcal{J} = \{\mathsf{GN}_3\}$

3.3 Remove$(\mathcal{I}, \mathcal{R})$

If soldiers die in the battlefield or leave the current group, then the remaining soldiers should regenerate a new group session key by using Remove algorithm for backward secrecy. MBN node first deletes ephemeral session keys and related information such as passwords and names of the leaving GN nodes. A MBN node generates keying materials and broadcasts them to the remaining GN nodes, as illustrated in Figure 5. The Remove$(\mathcal{I}, \mathcal{R})$ algorithm takes inputs as an initial set $\mathcal{I}$ and a revoked set $\mathcal{R}$. Without loss of generality, we assume that $\mathcal{I} = \{\mathsf{GN}_1, .., \mathsf{GN}_n\}$ and $\mathcal{R} = \{\mathsf{GN}_{n-j}, ..., \mathsf{GN}_n\}$. Remove$(\mathcal{I}, \mathcal{R})$

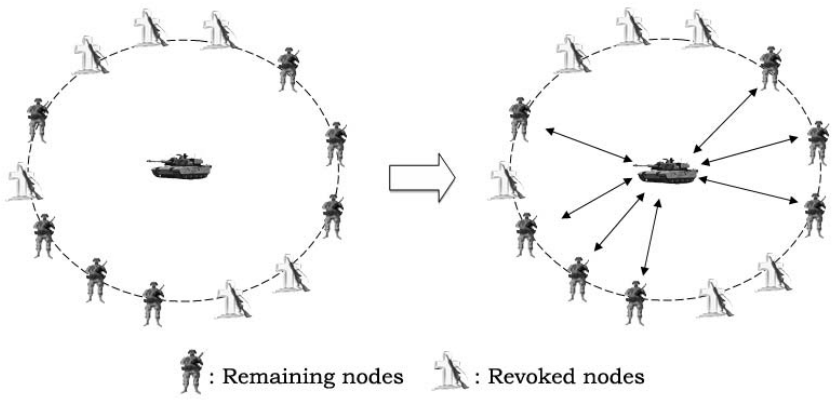


Fig. 5. The framework of Revoke Algorithm

	MBN	GN$_1$	GN$_2$	GN$_3$	GN$_4$
Setup$^1(\mathcal{J})$	$s_i \leftarrow [1, q-1]$ $\mathcal{E}_{pw_i}(g^{s_i})$	$x_1 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_1}(g^{x_1})$	$x_2 \leftarrow [1, q-1]$ $\mathcal{E}_{pw_2}(g^{x_2})$	$\overline{sk}_3$	$\overline{sk}_4$
SerCom$(\mathcal{J})$	$\mathtt{List}_{sc} = \{\overline{sk}_1, \overline{sk}_2\}$				
Setup$^2(\mathcal{J})$	$\boxed{\mathcal{H}_2(\overline{sk}_i \| S)}$	$\boxed{\mathcal{H}_2(\overline{sk}_1 \| \mathsf{GN}_1)}$	$\boxed{\mathcal{H}_2(\overline{sk}_2 \| \mathsf{GN}_2)}$		
	$N \leftarrow [1, q-1]$ $sk_1 \oplus N \| sk_2 \oplus N$	sk	sk		
NodeCom$(\mathcal{J})$		$\overline{sk}_1$	$\overline{sk}_2$		

Fig. 6. Revoke Algorithm of N-party EKE-D for $\mathcal{I} = \{\mathsf{GN}_1, \mathsf{GN}_2, \mathsf{GN}_3, \mathsf{GN}_4\}$, $\mathcal{R} = \{\mathsf{GN}_3, \mathsf{GN}_4\}$

algorithm first updates $I = I \setminus R$ and invokes $\mathsf{Setup}(\mathcal{I})$ algorithm. After finishing four sub-algorithms $\mathsf{Setup}^1(\mathcal{I})$, $\mathsf{SerCom}(\mathcal{I})$, $\mathsf{Setup}^2(\mathcal{I})$, and $\mathsf{NodeCom}(\mathcal{I})$, GN_i generates a new session key $sk = \mathcal{H}_3(SIDS||N)$ where N is a random value from Z_q^*, $SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||...||sk_{n-j+1} \oplus N$, and $sid' = \mathcal{E}_{pw_1}(g^{x_1})||\mathcal{E}_{pw_2}(g^{x_2})||...||\mathcal{E}_{pw_{n-j+1}}(g^{x_{n-j+1}})$. For $\mathcal{I} = \{\mathsf{GN}_1, \mathsf{GN}_2, \mathsf{GN}_3, \mathsf{GN}_4\}$, $\mathcal{R} = \{\mathsf{GN}_3, \mathsf{GN}_4\}$, we illustrate $\mathsf{Revoke}(\mathcal{I}, \mathcal{R})$ algorithm in Figure 6.

4 Security Analysis

In this section we prove that an N-party EKE-D protocol achieves the NEKED security conditions under the well-known computational assumptions. It is clear that the protocol satisfies the **Completeness** property. First, we prove that an advantage of a session key that an adversary $\mathcal{A}$ tries to get is negligible under the computational *Diffie-Hellman* assumption. The security theorem is as follows.

4.1 Indistinguishability

Theorem 3.1. *P is an N-party EKE-D protocol of Figure 2, and passwords are chosen from a finite dictionary of size $|\mathcal{D}|$. Let $\mathcal{A}$ be a probabilistic polynomial time adversary which asks q_s send, q_h hash, and $q_\mathcal{E}$ encryption/decryption queries. Then*

$$\mathbf{Adv}_P^{neked}(\mathcal{A}) \leq \frac{q_\mathcal{E}^2 + q_{h_1}^2 + 2q_{h_2}^2 + q_{h_3}^2}{(q-1)} + \frac{2q_s}{|\mathcal{D}|} + 2q_{h_1} \cdot \mathbf{Adv}_\mathbb{G}^{cdh}(T_\Delta)$$

where T_Δ is polynomial running time of **CDH** *adversary algorithm $\mathcal{A}_\Delta$ such that $T_\Delta \leq T + (q_s + q_\mathcal{E})(\tau_G + \tau_\mathcal{E})$. τ_G and $\tau_\mathcal{E}$ are computational time for exponentiation and encryption. q is the order of G.*

Proof. We define several experiments from a real experiment $\boldsymbol{Exp}_0$ to $\boldsymbol{Exp}_3$. In each experiment, we simulate various behaviors of an adversary and then measure the advantage of an adversary on the session key. That is, in each experiment $\boldsymbol{Exp}_i$, when an adversary $\mathcal{A}$ asks a Test query, a coin b is flipped. Then $\mathcal{A}$ guesses b, and outputs the guessed bit b'. Let $\mathbf{Succ}_\mathcal{A}^i$ be the event that $b' = b$ in the $\boldsymbol{Exp}_i$ and $Pr[\mathbf{Succ}_\mathcal{A}^i]$ be its probability. At the end of experiments, we measure the difference of probability, $Pr[\mathbf{Succ}_\mathcal{A}^i] - Pr[\mathbf{Succ}_\mathcal{A}^{i+1}]$, between $\boldsymbol{Exp}_i$ and $\boldsymbol{Exp}_{i+1}$ for $0 \leq i \leq 1$. By using the probability we finally get the result of **Theorem 3.1**. The proof of this theorem will appear in full version of the paper.

4.2 Forward and Backward Secrecy

In the next theorems, we show that an N-party EKE-D protocol satisfies forward and backward secrecy.

Theorem 3.2. *An N-party EKE-D protocol P provides forward secrecy assuming that the computational Diffie-Hellman problem is hard in the group $\mathbb{G}$.*

Proof. By executing $\mathsf{Setup}^1(\mathcal{J})$ algorithm, newly joined nodes ($\in \mathcal{J} = \{n+1, n+2, .., n+j\}$) compute their own intermediate *Diffie-Hellman* keys $sk_l (n+1 \leq l \leq n+j)$. With the keys the new nodes can compute current (or future) group session keys. However, to make an old session key sk, the members must know past intermediate key sk_i. Computing sk_i is reduced to the problem of **CDH**. If the new members obtain the password of members in $\mathcal{I}$ by asking $\mathsf{Corrupt}$ queries, the newly joined nodes compute the old *Diffie-Hellman* instance g^{x_i}, g^{s_i}. But, they never compute sk_i without solving a **CDH** problem of $g^{x_i s_i}$. Without a formal proof, we can conclude that the newly joined group members can not compute old session keys because they can not solve a **CDH** problem. □

Theorem 3.3. *An N-party EKE-D protocol P provides backward secrecy assuming that (1) F is a collision resistant hash function and (2) the computational Diffie-Hellman problem is hard in the group* $\mathbb{G}$.

Proof. Let's suppose that $\mathcal{I}(=\{1, ..., n\})$ is the set of total members and $\mathcal{R}$ $(=\{n-j, n-j+1, ..., n\})$ is the set of revokes nodes (Namely, the remaining members are $\mathsf{GN}_1, ..., \mathsf{GN}_{n-j-1}$). By executing $\mathsf{Revoke}(\mathcal{I}, \mathcal{R})$ algorithm, the nodes of $\mathcal{I} = \mathcal{I} \setminus \mathcal{J}$ invoke $\mathsf{Setup}(\mathcal{I})$ algorithm with an input of $\mathcal{I}$. Four sub-algorithms $\mathsf{Setup}^1(\mathcal{J})$, $\mathsf{SerCom}(\mathcal{I})$, $\mathsf{Setup}^2(\mathcal{I})$, and $\mathsf{NodeCom}^2(\mathcal{I})$ are performed, and the remaining nodes generate a new session key.

The revoked nodes keep the intermediate keys $\overline{sk_i} = F(sk_i)$, but the nodes do not compute intermediate keys of the remaining members since they do not know passwords. In addition, since F is a collision resistant hash function the revoked nodes can not find valid collision keys from the intermediate keys of them. □

4.3 Insider Guessing Attacks

The above theorems do not guarantee the security against a guessing attack by a malicious insider. Let's assume a military spy in the group, who tries to collect high-grade information such as passwords. The goal of the spy would be how to collect passwords' information, undetectably. Generally, this insider attack is said to be an undetectable on-line guessing attack. Actually, the undetectable on-line guessing attack is first mentioned by Ding and Horster [14]. A malicious insider user first guesses a password *pw* of one of the users and uses his guess in an on-line transaction. The malicious user verifies correctness of his guess using responses of a server S. Note that a failed guess never be noticed by S and other users. Thus, the malicious user can get sufficient information on *pw* by participating the protocol legally and undetectably many times. Below we show that the suggested N-party EKE-D protocol is also secure against an undetectable on-line guessing attack.

Theorem 3.4. *An N-party EKE-D protocol P is secure against undetectable on-line dictionary attacks by malicious insider users.*

Proof. Let's $\mathcal{U}_k$ be a malicious soldier trying to guess a password pw_i of valid GN_i nodes for $1 \leq k \neq i \leq n$. To perform on-line guessing attack undetectably, $\mathcal{U}_k$ should generate a valid hash authenticator (HA) in the second round.

- First, let's consider the event that $\mathcal{U}_k$ computes the common session key, and then generates the valid HA. In the first round, $\mathcal{U}_k$ first impersonates GN_i, and broadcasts $\mathcal{E}_{pw'_i}(g^{x_i})$ to MBN by using an appropriate password pw'_i and a random value x_i. In the second round, $\mathcal{U}_k$ receives $\mathcal{E}_{pw_i}(g^{s_i})$ from MBN, and computes $\mathcal{D}_{pw'_i}(\mathcal{E}_{pw_i}(g^{s_i})) = g^{s'_i}$ for some element s'_i which is unknown to $\mathcal{U}_k$. Therefore, even if $\mathcal{U}_k$ knows x_i, $\mathcal{U}_k$ can not compute $sk_i(= \mathcal{H}_1(sid'||g^{x_i s_i}))$.
- Second, let's consider the event that $\mathcal{U}_k$ forges a valid HA without knowing sk_i. This event is exactly collusion event of output of HA. The hash in the protocol is an ideal hash function, hence the probability is bounded by $\frac{q^2}{2^l}$ according to the birthday paradox where $\mathcal{U}_k$ makes at most q hash queries.

By the above two cases, $\mathcal{U}_k$ cannot undetectably succeed an on-line guessing attack in N-party EKE-D protocol.

5 Summary and Further Works

In this paper, we proposed a password-based group key exchange with different passwords for the ground multi-layer mobile ad-hoc network. When the GN nodes leave and join, our proposed N-party EKE-D protocol only requires constant-round to re-generate a new group session key, hence it is scalable. We showed that the N-party EKE-D protocol is secure assuming that computational *Diffie-Hellman* problem is hard. We have the following future works.

- **General Security Model for Multi-layer MANET.** The security model considered in this paper is just for the ground MANET. We need to construct a general security model including second and third layers, and design a new efficient and secure key management protocol based on the general security model.
- **Cross-realm setting.** In this paper, we only considered group key exchange in the same MBN realm. In the real battlefields, an authentication setting that two GN nodes registered in different MBNs wants to communicate, securely, is arisen more often. It may be worth to design an authenticated key exchange protocol between two GN nodes in the different realms.
- **Generic construction.** It would be a good future work to design a generic construction of password-based authenticated group key exchange protocols under the dynamic setting based on any secure and efficient 2-party protocols. In this general construction, we can avoid the random oracle and ideal cipher models by using the existing efficient protocol based on standard assumption.

Acknowledgement

We thank Ik Rae Jeong for helpful discussions on an early version of this paper. We also thank anonymous referees of SPC 2006 for their valuable comments.

References

1. M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks", *In proceedings of Eurocrypt'00*, LNCS Vol.1807, pp. 139-155, Springer-Verlag, 2000.
2. M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *In proceedings of the First ACM Conference on Computer and Communications Security, ACM*, 1995
3. M. Bellare and P. Rogaway, "Entity authentication and key distribution", *In Proceedings of Crypto'93*, LNCS Vol. 773, pp. 232-249. Springer-Verlag, 1994
4. S. Bellovin and M. Merrit, "Encrypted key exchange: password based protocols secure against dictionary attacks", *In proceedings of the Symposium on Security and Privacy*, pp.72-84, IEEE, 1992.
5. E. Bresson, O. Chevassut, and D. Pointcheval, "Group diffie-hellman key exchange secure against dictionary attacks", *In proceedings of Asiacrypt'02*, LNCS Vol. 2501, pp. 497-514, Springer-Verlag, 2002.
6. E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater, "Provably authenticated group diffie-hellman key exchange", *In proceedings of 8th ACM Conference on Computer and Communications Security*, pp. 255-264, 2001.
7. J. Black and P. Rogaway, "Ciphers with arbitrary finite domains", *In proceedings of CT-RSA conference*, LNCS Vol. 2271, pp. 114-130, Springer-Verlag, 2001.
8. V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using diffie-hellman", *In proceedings of Eurocrypt'00*, LNCS Vol. 1807, pp. 156-171, Springer-Verlag, 2000.
9. J. W. Byun and D. H. Lee, "N-party Encrypted Diffie-Hellman Key Exchange Using Different Passwords", *In proceedings of ACNS05'*, LNCS Vol. 3531, page 75-90, Springer-Verlag, 2005.
10. J. W. Byun, I. R. Jeong, D. H. Lee, and C. Park, "Password-authenticated key exchange between clients with different passwords", *In proceedings of ICICS'02*, LNCS Vol. 2513, pp. 134-146, Springer-Verlag, 2002.
11. J. W. Byun, D. H. Lee, and J. Lim, "Password-based Group Key Exchange Secure Against Insider Guessing Attacks", *In proceedings of CIS'05*, LNAI Vol. 3802, pp. 143-148, Springer-Verlag, 2005.
12. J. W. Byun, D. H. Lee, and J. Lim, "Efficient and Provably Secure Client-to-Client Password-Based Key Exchange Protocol", *In proceedings of APWEB'06*, LNCS Vol. 3841, pp. 830-836, Springer-Verlag, 2006.
13. L. Chen, "A weakness of the password-authenticated key agreement between clients with different passwords scheme", ISO/IEC JTC 1/SC27 N3716.
14. Y. Ding and P. Horster, "Undetectable on-line password guessing attacks", *In ACM Operating Systems Review*, Vol. 29, No. 4, pp. 77-86, 1995.
15. O. Goldreich and Y. Lindell, "Session-key generation using human passwords only", *In proceedings of Crypto'01*, LNCS Vol. 2139, pp. 408-432, Springer-Verlag, 2001.
16. D. Gu, H. Ly, X. Hong, M. Gerla, G. Pei, and Y. Lee, "C-ICAMA: A Centralized Intelligent Channel Assigned Multiple Access for Multi-layer Ad-hoc Wireless Networks with UAVs", *In proceedings of IEEE WCNS'00*, pp. 879-884, 2000.
17. D. Gu, G. Pei, H. Ly, M. Gerla, and X. Hong, "Hierarchical Routing for Multi-layer Ad-hoc Wireless Networks with UAVs", *In proceedings of IEEE MILCOM'00*, pp. 310-314, 2000.
18. D. Gu, G. Pei, H. Ly, M. Gerla, B. Zhang, and X. Hong, "UAV-aided Intelligent Routing for Ad-hoc Wireless Network in Single-area Theater", *In proceedings of IEEE WCNS'00*, pp. 1220-1225, 2000.

19. S. Halevi and H. Krawczyk, “Public-key cryptography and password protcols”, *In proceedings ACM Conference on Computer and Communications Security*, ACM press, pp. 63-72, 1999.
20. D. Jablon, “Strong password-only authenticated key exchange”, *In Computer Communication Review*, Vol.26, No.5, pp. 5-26, 1996.
21. J. Katz, R. Ostrovsky, and M. Yung, “Efficient password-authenticated key exchange using human-memorable passwords”, *In proceedings of Eurocrypt’01*, LNCS Vol. 2045, pp. 475-494, Springer-Verlag, 2001.
22. Y. Kim, A. Perrig, and G. Tsudik, “Simple and fault-tolerant key agreement for dynamic collaborative groups”, *In proceedings of 7th ACM CCS’00*, pp. 235-244, 2000.
23. J. Kim, S. Kim, J. Kwak, and D. Won, “Cryptoanalysis and improvements of pass word authenticated key exchange scheme between clients with different passwords”, *In proceedings of ICCSA 2004*, LNCS Vol. 3044, pp. 895902, Springer-Verlag, 2004.
24. S. Lucks, “Open key exchange: how to defeat dictionary attacks without encryting public keys”, *In proceedings of the security protocol workshop ’97*, pp. 79-90, 1997.
25. R. C.-W. Phan and B. Goi, “Cryptanalysis of an Improved Client-to-Client Password-authenticated Key Exchange (C2C-PAKE) Scheme”, *In proceedings of ACNS 2005*, LNCS Vol. 3531, pp. 3339, Springer-Verlag, 2005.
26. K. Rhee, Y. Park, and G. Tsudik, “A Group Key Management Architecture for Mobile Ad-hoc Wireless Networks”, Jornal of information science and engineering 21. pp. 415-428, 2005.
27. V. Shoup, “OAEP reconsidered”, *In proceedings of Crypto01’*, LNCS Vol. 2139, pp. 239-259, 2001.
28. Q. Tang and L. Chen, “Weaknesses in two group Diffie-Hellman Key Exchange Protocols”, Cryptology ePrint Archive 2005/197, 2005.
29. T. Wu, “Secure remote password protocol”, *In proceedings of the Internet Society Network and Distributed System Security Symposium*, pp. 97-111, 1998.

Enabling Secure Discovery in a Pervasive Environment

Slim Trabelsi[1], Jean-Christophe Pazzaglia[1,2], and Yves Roudier[1]

[1] Institut Eurécom, 2229 Route des Crêtes - BP 193,
06904 Sophia Antipolis Cedex - France
{Slim.Trabelsi, Yves.Roudier}@eurecom.fr
[2] SAP Labs France, 805 Avenue du Dr Donat - BP 1216,
06254 Mougins Cedex - France
Jean-Christophe.Pazzaglia@sap.com

Abstract. The pervasive computing paradigm assumes an essentially dynamic model of interaction between devices that also motivates the need to discover the services offered by previously unknown parties at an early phase of these interactions. Whereas this assumption is at the heart of many pervasive computing protocols and systems, the necessity of securing service discovery and the complexity of this task have been largely underestimated, if considered at all. This paper discusses the implications of insecure service discovery in available systems and which security objectives should be pursued. The design space for introducing security features into a specific architecture, namely registry-based discovery systems, is then explored and assessed.

1 Introduction

The Service Oriented Architecture (SOA) paradigm, currently boasted by Web Services, was initially developed in the Jini [13] framework to address the specific requirements of pervasive computing software. In particular, SOA was originally intended to enable access to applications running on nearby devices in a dynamic fashion. This programming style promotes the use of loosely coupled and highly interoperable applications to overstep the limitations of traditional distributed component solutions (CORBA, DCOM). A *Service*, the building block of SOA solutions, is intended for encapsulating a set of related business functions within a container and for enabling access to these functions through standardized interfaces. In pervasive computing, context-awareness would typically be enabled through the access to a set of such services.

Orchestration techniques were developed in order to deploy a set of basic services as a more complex service. Even though static orchestration is frequently used, for example in Web Services based architectures, we argue that the quintessence of the SOA style, especially when used for pervasive computing software, lies in the dynamic composition of services. Such a dynamic composition obviously comes at a cost: being able to locate previously unknown services becomes mandatory.

Discovery therefore becomes of strategic importance in the SOA stack and this importance is growing proportionally with the dynamic aspect of the environment: while a typical intranet implementation may rely on a basic discovery strategy (e.g. naming service in CORBA) or may even not strictly require it (e.g. predefined set of known services), Internet-wide and, above all, pervasive applications face a set of challenges with respect to discovery. In such applications, the discovery strategy should cope with the heterogeneity of services and platforms from a technical perspective (e.g. take into account bandwidth, energy savings …), with the complex semantics of service descriptions (e.g. resorting to terminology- or ontology-based descriptions), with the scalability of the solution, and with the requirements for security and trust regarding the services discovered. Since the emergence of the SOA paradigm, different works (e.g. UDDI [14], WS-Discovery [3], OWL-S [6]) have aimed at solving many of these issues, yet only very few of them address security and trust, which are however essential to the successful deployment of pervasive computing solutions. For instance, services should protect sensitive information as well as their availability from rogue users; and private information of a user should not be revealed to a service without assessing that service's potential maliciousness.

This paper is organized as follows. Section 2 addresses the relationships between security and service discovery mechanisms. Section 3 discusses different designs that can be incorporated into existent protocols in order to secure them. Section 4 details the security features that should be introduced into a registry based discovery protocol and their handling. Section 5 compares the service discovery model explored in this paper with related work. Open issues not addressed in this paper are finally discussed in conclusion.

2 Service Discovery and Security

A common misconception is to limit service discovery to the registry[1] supported architecture that many standard SOA based services (like for instance Jini's reggie, UDDI, etc.) have adopted de facto in their implementations. In such solutions, as illustrated by Figure 1, the registry is responsible for putting in touch services and their clients. A service advertises its capabilities to the registry, which will store them for a certain amount of time. A client solicits the registry to find a service by sending a request containing service preferences, which the registry tries to match with the most suitable provider found from the stored advertisements. This now rather conventional approach to SOA introduces a third party, which clearly must be trusted.

Service discovery may alternatively rely on a peer-to-peer architecture. Every client then can perform a service discovery by broadcasting its request to its neighborhood, and if one of the neighbors is able to fulfill the request, it will send a response to the requester. The neighbor may otherwise forward this request to its own neighborhood. This mechanism is used for instance by the P2P-based Web Service Discovery system (PWSD) [15], which relies on the Chord P2P protocol to perform the service discovery. The rest of this paper essentially discusses the registry based model, also called service discovery service.

[1] To be more precise, this solution uses a middle-agent mechanism often called registry, depending on its capabilities, this middle-agent is also referred in the literature under the names of directory, discovery service, broker, facilitator,…

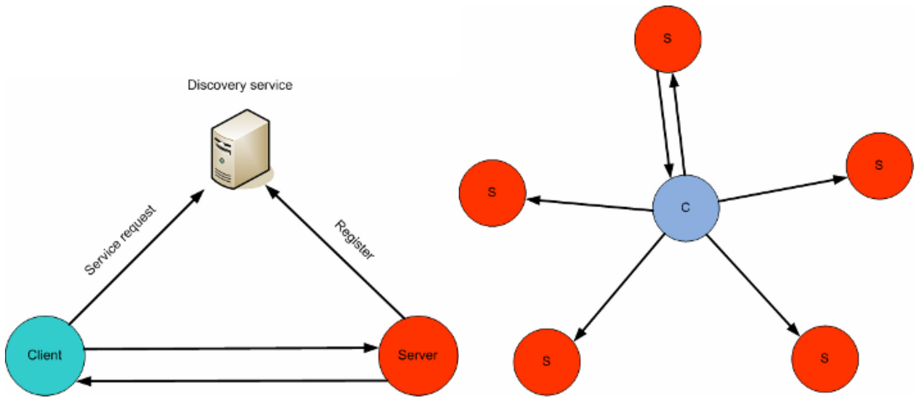


Fig. 1. Middle Agent Service vs. P2P Discovery Architecture

2.1 Revisiting Security Threats for Service Discovery

It should be underlined that the principal actors of the discovery phase are indeed the service requester and the service provider, even in the case of a registry based service discovery. The main specificity of discovery is that, by definition, these actors are initially unaware of their respective existence and of their security policies. In addition, they are likely members of different administrative domains. Discovery is very often at the initiative of the service requester (e.g. *lookup model*) but can also be initiated by the service provider (e.g. *advert model)*. The initiator takes a more important risk than the other party since it does not *control which entities will receive* the discovery message, nor the potential usage of the information embedded in this message. This peculiar situation raises two threats:

- *Client and Service Authentication*: The very objective of service discovery is to communicate with previously unknown entities that provide specific functionalities. Open discovery services therefore require that the first message sent (*lookup* or *advert*) is in clear, also meaning that the content of the message can be accessed. Without the means to authenticate clients and servers, service discovery makes the implementation of a man-in-the-middle attack possible [11], a malicious entity being able to wrongly answer a discovery message. Registry based discovery schemes make it much simpler than infrastructure-less ones to perform secure discoveries, since the registry is the only element which the client needs to identify and which it should be identified from, thereafter enabling message protection through their encryption or integrity check.
- *Privacy*: In the *lookup* model, the information disclosed in the request is likely to reveal a subset of the intentions of the service requester. Similarly, service providers may want to avoid potential commercial competitor or malware to gather information about their offers too easily. In both cases, attaching an identity certificate to enable the proper identification of parties in presence and the protection of subsequent messages will only expose more information (name, address …). The correlation of such information with discovery related information is particularly worrying from a privacy protection perspective.

In addition to the previous two threats that are relatively specific, the overall purpose of service discovery is to access some resources. This yields more classical threats:

- *Access control*: Since client/service authentication is problematic in the initial discovery phase, traditional service oriented architectures do not support access control during the discovery phase. Service providers would ideally advertise their services exclusively to potential users, even though this objective is in practice difficult to achieve in a pervasive environment. Disclosing the description of a service to any requester potentially increases the risk that a malicious user take advantage of this knowledge to access restricted services.
- *Availability*: Denial of Service (DoS) is an attack to the availability of resources preventing the authorized access to a system resource or delaying system operations and functions. Openly exposing service descriptions during discovery enables attackers to exploit vulnerabilities by creating specially crafted messages for the server or by the registry. Notably, registries clearly constitute a single point of failure and therefore are particularly sensitive to brute force DoS attacks.

2.2 Objectives

With the success of Web Services, the momentum behind Service Oriented Architecture is enormous and thousands of services are already available on the Internet. Similarly, the deployment of wireless networks and small devices (sensors, mobile phones, PDAs, RFIDs, etc.) is likely to boost the startup of local services, thereby paving the way for the actual deployment of pervasive computing systems. In such landscape, where mobility is the usage, clients and servers should protect themselves from malicious software, including at discovery, the very first step of any interaction.

Clients should be able to find a service matching their preferences, these preferences encompassing different characteristics of the service: the service functional definition *per se*, its cost, its quality, as well as the security and privacy requirements imposed by the service (e.g. encryption strength). On the client side, the user should be certain that only services matching his preferences would be chosen and later contacted: from his point of view, trusting a service should therefore go beyond the simple authentication of the service provider and also establish the veracity of the features exposed by the service.

On the server side, the problem is quite similar since the server does not know the users that can potentially gain access to its service. Services should therefore be accessible only if they trust the client to access them according to a precise policy that protects them. Matching this policy description is therefore an important part of the discovery process.

Several important concerns should therefore be taken into account in order to enable a secure service discovery service:

- *Security policies for service discovery*
  The different entities participating to service discovery each specify their own discovery objectives as policies. Security policies should be a part of these objectives and should be described using a language common to all entities. Ideally, this language should be expressive and flexible enough to describe

standard operations intended to ensure the security of discovery (authentication means, access control, encryption level, etc.) as well as more advanced mechanisms such as auditing or privacy related issues. The matching of client and server discovery policies in general, and security policies in particular, is likely to use reasoning techniques from the Semantic Web area.

- *Responsibility for enforcing the discovery policy*
  Discovery policies can be enforced by the client and by the server. However, discovery policies may also be sent remotely and their enforcement can be delegated to a third party trusted by the client and the service, like the registry. This notably implies that the semantics attached to the policy is non ambiguous.
- *Limiting data exposure during the service discovery process*
  Discovering a service may imply that the user disclose his identity, the kind of service he is looking for, and other personal information like certificates or his location. Even if such information were mainly public, it might contain some data that should not be disclosed to everyone. In order to protect the privacy of the user, a mechanism should guarantee the diffusion and retention of these data. If a service is publicly available and publicized, it will also be more exposed to attacks. A possible way to prevent these attacks is to hide the service, or more precisely to delay its discovery. Comparing services to houses and service methods to doors, it will be far more difficult for a thief to illegally access the house if the doors are hidden and if he does not have any idea about the mechanisms used to open these doors (code, locks …). Servers should similarly have the possibility to choose who can discover their services.
- *Establishing trustworthy relationships*
  In pervasive environments, devices and users are likely belonging to different administrative domains. Consequently, an entity cannot take for granted that a discovery related information is true without appropriate proofs. Building secure federations of devices may be required to establish trust on a longer term basis, using extended attribute certificates or even incentive systems (reputation, monetary incentives …).

Next section focuses on describing the adaptation of a registry based protocol enabling it to take into account these concerns and objectives.

3 Secure Service Discovery Model

Assigning to a trusted entity of the system the responsibility to enforce the discovery policies and in particular their security part as defined by users is critical to service discovery. To avoid raising the complexity of service discovery, we do not propose to add a new entity to SOA architectures together with a dedicated protocol, but rather to assign this task to the registry. The reason for choosing the registry as a policy enforcement point is that the users of the system typically already need to trust this entity. This solution thereby implements a distributed security policy and assumes that servers and clients can communicate with the registry through the use of a PKI. (e.g. knowledge of the registry's public certificate) or through some more advanced trust establishment mechanism (which may be more adapted to pervasive computing, see for instance [2]).

3.1 Secure Discovery Service Use Cases

Before detailing the protocol, the policies expressiveness, and the registry capabilities, we will explain in the following use cases how the registry behavior and the messages are modified to fulfill the security requirements imposed by the entities involved in discovery activity. To illustrate these different needs, we take the example of different services offered within a research laboratory. Let us first assume the availability of printing services, whose access should be controlled or monitored due to funding constraints.

The first printing service is a standard black and white printer that is accessible to everybody. This service is advertised to the registry by sending the service description without imposing any particular security constraint. The client will simply send a message containing a service template in order to acquire the exact service description from the registry. This example, shown in Figure 2, introduces the basic discovery protocol, largely inspired by Jini, that we assume to exist and reason about.

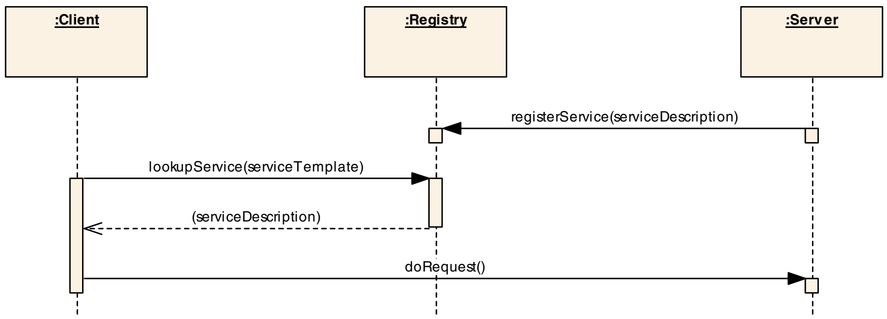

:Client
:Registry
:Server
registerService(serviceDescription)
lookupService(serviceTemplate)
(serviceDescription)
doRequest()

Fig. 2. Basic Service Discovery: the Black and White Printing Service

The second printing service involves a color printer only accessible by the laboratory staff and their guests. The registration message sent by the server should include a security policy that specifies that the client must be authenticated by the registry and that he must provide a discovery proof signed by the registry to access the service. The client sends his lookup service request by specifying a service template and, in order to authenticate itself, an identity certificate[2]. The registry verifies if the user pertains to the staff or if he is a trusted guest, and then generates a discovery proof (for example a signed SAML assertion). In order to print, the client will have to exhibit this discovery proof within the request as shown in Figure 3.

[2] **NB:** Certificates and proofs are shown as constraints in the UML diagram to provide a condensed representation. These constraints should be enforced through network layer (message over SSL) or using encryption, signature, or metadata embedded in the message (for example SOAP using WSS, XML-DSig, SAML assertions …).

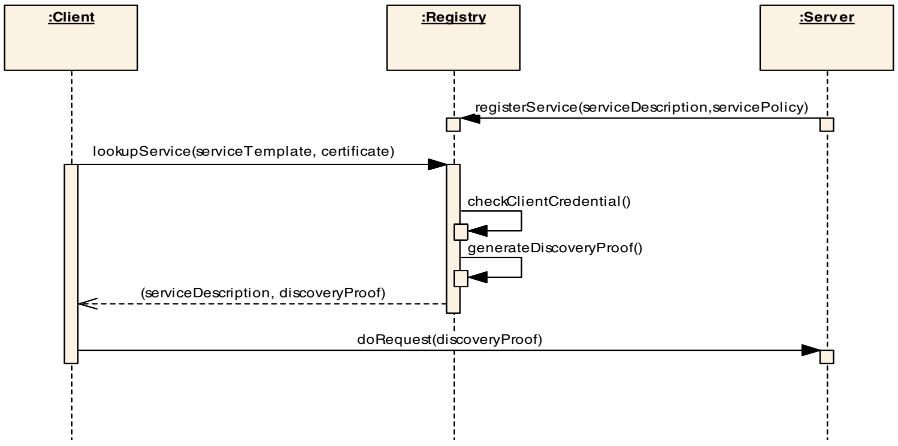

:Client
:Registry
:Server
registerService(serviceDescription,servicePolicy)
lookupService(serviceTemplate, certificate)
checkClientCredential()
generateDiscoveryProof()
(serviceDescription, discoveryProof)
doRequest(discoveryProof)

Fig. 3. Secure Service Discovery: the Color Printing Service

The third printing service, a photo printer, is restricted to the Multimedia research team staff. The registration phase is similar to the color printer example: the server advertises itself to the registry by sending a message containing the service description, its security policy requesting the registry to authenticate clients, and also its certificate in order to be authenticated by the registry. The client lookup request contains a service template and a security policy. This policy specifies that the registry has to authenticate the server (to protect the client), while the server policy requires generating a traceable token as shown in Figure 4. This token (for example a SPKI certificate) will be used by the server to authenticate the client while preserving

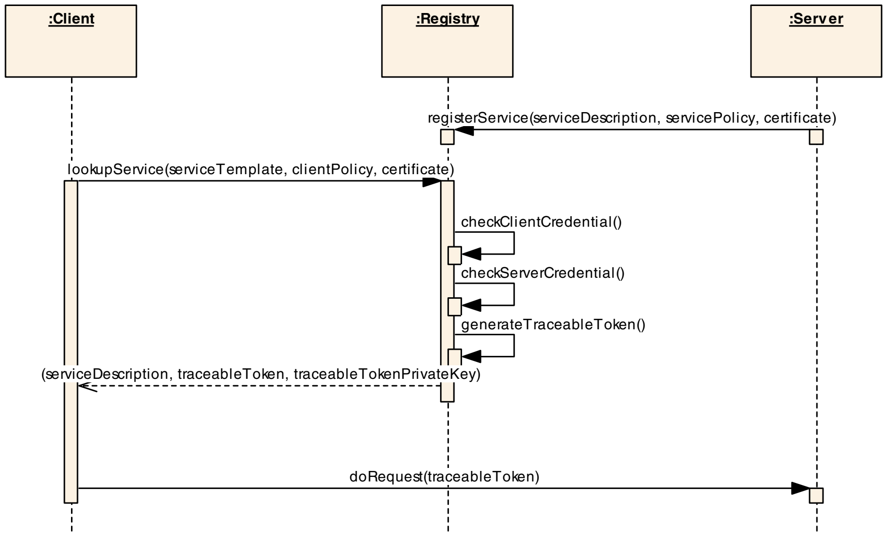

:Client
:Registry
:Server
registerService(serviceDescription, servicePolicy, certificate)
lookupService(serviceTemplate, clientPolicy, certificate)
checkClientCredential()
checkServerCredential()
generateTraceableToken()
(serviceDescription, traceableToken, traceableTokenPrivateKey)
doRequest(traceableToken)

Fig. 4. Secure Service Discovery: the Photo Printing Service

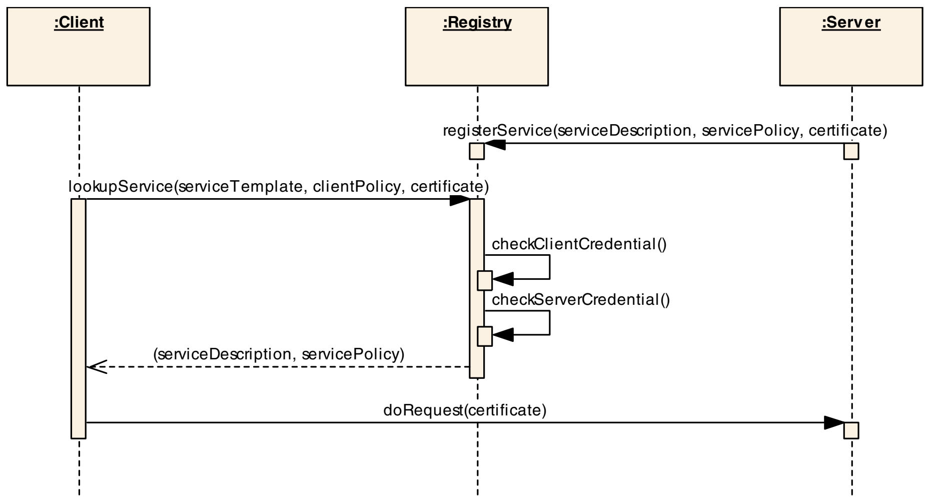

:Client
:Registry
:Server
registerService(serviceDescription, servicePolicy, certificate)
lookupService(serviceTemplate, clientPolicy, certificate)
checkClientCredential()
checkServerCredential()
(serviceDescription, servicePolicy)
doRequest(certificate)

Fig. 5. Secure Service Discovery: the File Sharing Service

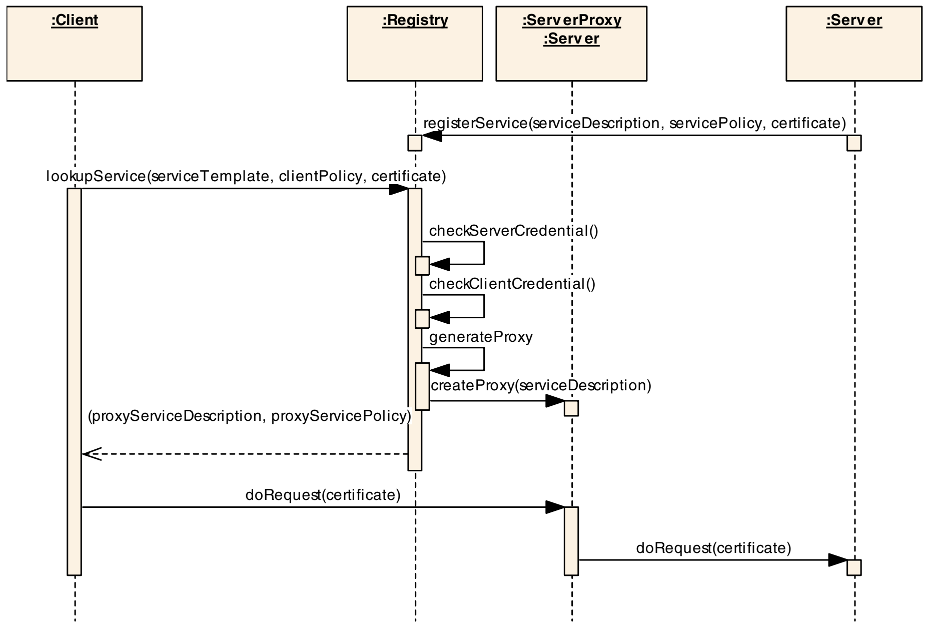

:Client
:Registry
:ServerProxy :Server
:Server
registerService(serviceDescription, servicePolicy, certificate)
lookupService(serviceTemplate, clientPolicy, certificate)
checkServerCredential()
checkClientCredential()
generateProxy
createProxy(serviceDescription)
(proxyServiceDescription, proxyServicePolicy)
doRequest(certificate)
doRequest(certificate)

Fig. 6. Secure Service Discovery: the Backup Service

his privacy, yet under certain circumstances, an authorized third party would be able to find the client's identity by contacting the registry.

In addition to the printing services, the laboratory also gives users access to two other services: a file sharing service and a backup service. In order to manage file access rights, both services require authenticating the users. In order to enable access to these services, the registry will check that the client belongs to the right user group and also that he is willing to disclose his identity by means of its digital certificate.

The backup service is working quite similarly to the file sharing service. For security reasons however, the server will not grant access to a client belonging to guests and managed by an external administrator. The access still can be granted provided that the registry generates a temporary proxy within the laboratory's administrative domain. Therefore, instead of sending the service description to the client, the registry will generate a proxy and send the proxy service description. The client will send his request to the proxy which will forward it to the server. In other circumstances, the proxy generation may also be triggered by the client in order to better protect its privacy. In this case, the proxy could authenticate the clients but only exhibit traceable credentials to the server like in the photo printing services described before.

4 Detailing the Protocol and Registry

Standard service discovery schemes do not enable the clients and the servers to impose their own security policy during the *lookup* phase. The originality of our solution is that we consider that service discovery transactions have to satisfy the security preferences of both entities by design. To fulfill this requirement, each client and server should export its security policy using a common language with well-defined and non ambiguous semantics. As mentioned earlier, the registry must be trusted by all the entities of the system since the security policy enforcement during discovery will be delegated to it. As we rely on a PKI infrastructure to authenticate this registry in our examples, the different clients and services must know the registry's public key to protect discovery messages.

4.1 Security Levels in the Discovery Policy

Broadly speaking, security policies are meant to define the reciprocal commitments of the partners involved in a specific process. Their coverage usually depends on the system security objectives (confidentiality, privacy, access control, availability…) and on the degree of expressiveness of the security policy language (e.g. from a low level of expressiveness like with WS-Policy [5] to a high level like with Rei [4]). In the case of service discovery, the scope of the policies may vary: they can be restricted to the service discovery process or they can impose constraints to the service execution or access. Three levels of policy scopes should be distinguished:

- Level 0: No security policy definition; the user must have the possibility to perform a simple and non secure service discovery. In that case, the different entities do not need to specify a particular security policy.
- Level 1: A security policy restricted to the service discovery aspects. The servers set up a security policy to limit their visibility to some restricted clients. Symmetrically, the clients may limit the scope of diffusion of their request to some particular servers. These policies will be enforced by the registry and they do not apply to the future transactions between the client and the service.

- Level 2: A security policy that specifies requirements of the service discovery step and also specifies how to configure access control to the service. Indeed, being able to discover a service does not necessary imply being able to effectively, even less securely access this service if one does not know the proper requirements. The policy rules can also force entities to set security associations (like the choice of the encryption key size or the encryption algorithm) or may imply that the client exhibit generic or specific credentials generated during the service discovery process. A level 2 policy may specify that the registry should take care of forwarding service requests after service discovery because of privacy requirements in the policy.

To our knowledge, no security policy language specifically is adapted to service discovery or enables expressing foreseen privacy or traceability requirements, even though inspiration might be taken out of e.g. EPAL or P3P. In our model, the registry primarily acts as a remote policy decision point (ensuring that client and server policies are compatible) but will also ensure the enforcement of privacy or traceability requirements. Still, the client or server may alternately implement and commit to the enforcement of other counterpart requirements or leave it to the registry. Level 2 policies also imply a tight integration with service access security mechanisms.

4.2 Service Registration: registerService

In order to register its service, the service provider advertises its capabilities to the registry. A `registerService` message should embed the following information:

- Service properties: this field contains the description of the service capabilities. This description will be used during the matchmaking process while looking for a suitable service provider for a specific client.
- A security policy: this field contains all the security preferences of the server. It can be considered as a set of access control rules that must be satisfied by the client to discover and access its proposed services. Such policies may be exchanged by using a standard protocol for metadata exchange (e.g. WS-MetadataExchange).
- A certificate: this certificate is signed by a trusted certification authority and contains certified information about the server (identity, public key, the lifetime of the service, recommendations …). The information embedded in this certificate can be used by the registry to verify that the server complies with the security constraints imposed by the service requestor.

The second and third components of the message are optional and vary with the expected security level. A service is public by default: if the server does not have the need nor the capabilities to protect its services, it does not have the obligation to specify a security policy. The server's certificate presence is optional for service discovery, but mandatory to enable its secure discovery. Moreover, its absence may restrict the potential constraints that clients may impose on the server's credentials or attributes described in the certificate.

4.3 Service Discovery: lookupService

In order to obtain a list of suitable services, the client must describe its service preferences and provide its credentials. Preferences will be sent to the registry, which will search for the services fitting with these requirements. The `lookupService` message should thus contain:

- Service template: this template is based on a partial instantiation of a service description constraining certain aspects of the service. This template may correspond with some existing services assuming that all the entities share a common taxonomy for instance. The registry would then match this template with the stored description of available services.
- A security policy: this policy contains the security constraints required by the client. These constraints describe the properties that must be satisfied by the service provider. These constraints may impose to enforce server authentication, to choose servers from a restricted set of administrative domains, or to check that the server respect precise rules concerning data retention, for example.
- A certificate: this certificate should be signed by a trusted certification authority and should contain certified information about the client (identity, public key, role…). This certificate can be used by the registry to verify that the client corresponds to the security criteria of a server that seems to match a lookup query.

Similarly to service registration, the policy and certificate are optional. This will obviously impose symmetrical limitations on the client's ability to locate services.

4.4 The Registry Matching and Policy Enforcement

The discovery process is initiated by the client sending a `lookupService` message to the registry. The registry will perform the following actions:

- Trust establishment and public key distribution: This bootstraps secure service discovery. The registry is considered as a trusted third party, and in order to establish this trust every entity that joins the system must acquire the public key certificate of the registry to identify it and to secure their communication.
- Match the service template contained in the client's message with the description of the registered services.
- Filter out the services that advertise a security policy incompatible with the client credentials or incompatible with the client's security policy.

In order to conform to the server and client policies, the registry may generate a suitable token such as a proof of discovery, an authentication receipt, or a temporary certificate, and associated keys to access the service. Finally, the registry returns the client a list of service descriptions answering his request, possibly with the token and server security policy that must be complied with to access to the proposed services.

4.5 Model Discussion

As explained in Section 2, challenges in securing service discovery cover the broad spectrum of security issues. Integrating security policies on top of a registry-based architecture solves these following concerns in this way:

- Authentication: each entity may require the other entities to authenticate themselves to the registry during the registry/discovery phase. The description of required authentication means during service delivery can also be specified using a level 2 policy (used during service matching and enforced later).

- Confidentiality: by using a PKI infrastructure to secure the transactions between the registries and other entities and by also using the security tokens to secure the communications between entities, only authorized entities can access messages.
- Access control: Server resources are protected against an unauthorized discovery thanks to security policies; furthermore the registry is able to enforce clients' requirements during service discovery by selecting only services providing valid credentials or secure enough access mechanisms.
- Trust: a PKI based solution establishes a trustworthy relationship with the registry. This trust relies on standard certification and trust chain. Further trust assessment will be performed by the registry itself. Although this can be seen as a limitation in fully ubiquitous environment, this solution is sufficiently flexible to enable the mobility of users across different administrative domains.
- Privacy: each entity can selectively expose its services to a restricted set of entities while remaining hidden for the others. Contrarily to existing solutions, our model does not focus only on servers, but also on clients and their requirements.
- Non repudiation: thanks to security tokens generated by the registry, an entity has the possibility to prove that it discovered the system legitimately. Access to this proof may also be controlled by policies to comply with a legislative framework.

This solution is strongly dependent on the existence of a trusted authority that is in charge of enforcing the security policies of the different entities in the system. To establish this trusted relationship, we also make an assumption concerning the *a priori* knowledge of the registry's identity or public key (or any kind of certificate) by services requestors and providers. These two constraints make the solution difficult to deploy in some conditions (the registry is missing or not reachable, or the users cannot securely acquire the registry certificate).

5 Related Work

Discovery-enabled pervasive computing systems generally address access control when a service is accessed, and make no restriction whatsoever regarding service discovery itself. In addition to privacy issues, this approach requires the application programmer to understand and address security issues at the implementation level instead of specifying an abstract and probably less error-prone security policy.

In contrast, [7] details the architecture of a service discovery service that addresses security at the discovery stage, as discussed in this paper. This work focuses on protecting the service side against malicious parties through strong authentication of end-points and the client side against private information disclosure and man-in-the-middle attacks through authentication and encryption. The system also supports the specification of a rudimentary discovery policy making it possible for a client to restrict the discovery to the set of services run by a trusted entity. However, availability and entity privacy are not addressed in this work.

The authors of [9] propose an architecture for secure service discovery. Components share a multicast address that will be used to bootstrap the communication. Directories (i.e. registries) use this multicast address to periodically announce their unicast address and certificate. Proxies are used to protect the servers by handling the registration, authentication, authorization, and key management for them, entities in the system setting up a session using hybrid encryption. The number of proxies (one by service) and heavy PKI infrastructure for securing the communication are likely to generate an important message overhead. Contrary to the claimed objective of this work to address pervasive systems, services are likely to be static, whereas the approach we advocate only requires the availability of a local registry, each client potentially being a server for other clients.

[12] considers privacy issues during service discovery. The proposal is based on the use of bloom filters to protect the client and server personal information (identity, certificates, attributes…) during directory and client authentication. However, the participants must agree in advance on the hash functions used. In comparison, the present paper's approach to privacy is quite different since the scenario it addresses relies on the existence of a trusted party for service discovery, whose role is only slightly extended to the non-disclosure of private information.

SPDP [8] is a discovery protocol for infrastructure-less and in particular registry-less systems, a much more complex situation than the one addressed in this paper. The design of this protocol is also strongly influenced by practical considerations regarding the electric consumption of mobile devices. SPDP however relies on a web-of-trust model to decide which devices may participate to the network and consequently to the discovery. This model however cannot delegate the enforcement of a discovery policy. In comparison, the approach presented in this paper relies on a TTP and a PKI, yet it still preserves a reasonable openness, if one considers that the scenarios depicted generally assume the availability of a trusted and known entity.

6 Conclusion

Service oriented architectures were originally introduced to address the deployment of software in a pervasive computing context. Even though they have been studied for some time now, the security of service discovery, one of their essential features, is generally weak. This paper analyzed such threats in SOA stacks that use a registry supported discovery and showed that discovery exhibits specific security needs, notably due to its early occurrence in the interaction of two devices. We proposed to reuse the messages of a standard discovery protocol by only extending their contents with the security information required to secure discovery and by delegating the handling of such information to the registry.

The solution proposed aims at dynamically configuring the mechanisms for protecting the security and privacy of the server and of the service requestor. It requires a trusted infrastructure for discovery, namely the registry, which plays a key role by ensuring that the discovery policies specified by the service and by the requestor are matched in a trustworthy manner. This prerequisite may limit the usage of the discovery mechanism in completely ad-hoc situations yet it is well adapted to the deployment of pervasive applications at airports, public agencies’ or private companies’ premises for instance. The discovery policies mentioned above raise some open issues regarding the fusion between client and service requirements: Semantic

Web mechanisms for service description and matchmaking [6] will probably need to be adapted in order to be able to describe and negotiate security aspects of policies.

We are currently implementing the solution described above as part of the middleware developed within the European project MOSQUITO[3] [10]. In addition to that solution, we are currently investigating an alternative infrastructure-less solution for service discovery which should still provide a significant subset of the security properties enforced by registry based service discovery.

Bibliography

1. F. Zhu, M. Mutka, and L. Ni "Classification of Service Discovery in Pervasive Computing Environments," MSU-CSE-02-24, Michigan State university, East Lansing, 2002.
2. L. Bussard, Y. Roudier "Embedding Distance Bounding Protocols within Intuitive Interactions" 1st International Conference on Security in Pervasive Computing, SPC'2003, Boppard, Germany, March 12-13, 2003.
3. OASIS, "WS-Discovery" http://msdn.microsoft.com/ws/2004/10/ws-discovery/
4. L. Kagal "Rei: A Policy Language for the Me-Centric Project" HP Technical Report, 2002
5. OASIS, "WS-Policy Specifications" http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policy1202.asp
6. D. Martin et al, "Bringing Semantics to Web Services: The OWL-S Approach", Proceedings of the 1st SWSWPC, USA 2004.
7. S. Czerwinski, et al "An architecture for a Secure Service Discovery Service" Proceedings of Mobicom'99, Seattle, USA, 1999
8. F. Almenarez, C. Campo: "SPDP: A Secure Service Discovery Protocol for Ad-hoc Networks", In 9th Open European Summer School and IFIP Workshop on Next Generation Networks, Budapest, 2003.
9. F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services", in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (Percom'03)*. IEEE Computer Society, Mar. 2003, pp. 235–242.
10. MOSQUITO Project IST 004636 https://www.mosquito-online.org/
11. M. Ghader *et al* "Secure resource and service discovery in personal networks" Wireless World Research Forum Meeting #12, Canada, 4-5 Nov 2004
12. F. Zhu, M. Mutka, L. Ni "Prudent exposure: A private and user centric service discovery protocol" Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04) Orlando, USA, 2004
13. SUN Microsystems, Jini Specifications http://java.sun.com/products/jini/
14. OASIS, "UDDI", http://www.uddi.org
15. O. Garofalakis et al "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?" 15th ACM Conference on Hypertext and Hypermedia (Hypertext 2004), Santa Cruz, USA, 2004

[3] The FP6 IST project MOSQUITO is supported by the European Community. This document does not represent the opinion of the European Community. It is also the sole responsibility of the author and not the responsibility of the European Community using any data that might appear therein.

Forward Secure Communication in Wireless Sensor Networks

Sjouke Mauw[1], Ivo van Vessem[1], and Bert Bos[2]

[1] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2] Chess Information Technology BV, De Ronde 15B, P.O. Box 273, 5680 AG Best, The Netherlands

Abstract. We propose a set of security provisions for node to base station communication in wireless sensor networks. It supports standard security requirements, viz. authentication of the origin of data and confidentiality of data. Additionally we use key evolution to achieve forward security which is of particular importance in the face of node capture attacks. As a bonus we obtain implicit weak freshness without message expansion. We take the typical resource constraints of wireless sensor networks into account. The security provisions can be superimposed on several communication models, such as the epidemic communication model.

1 Introduction

The main application of Wireless Sensor Networks is in monitoring the condition of a large area by distributing a collection of communicating sensors. Such networks are envisaged to consist of thousands of sensor nodes. In this scenario the budget per sensor is severely limited, which translates to a limited amount of chip area and a low capacity energy supply. This in turn puts tight restrictions on the amount of available storage, the affordable computational complexity, the amount of data that can be transmitted, and the transmission range.

Yet, there is an obvious need for authenticating sensor readings and some scenarios call for confidentiality as well. Consider, for example, the case of smoke detection sensors where an attacker could trigger false fire alarms if there were no means to establish the origin of a message. Sensors used for military purposes are also a prime target for adversarial manipulation. In a scenario where sensors monitor the presence of persons or perhaps their health condition, a confidential information exchange is required for privacy protection.

Our problem setting is characterised as follows. An area has to be monitored with respect to certain environment variables. For this purpose, a collection of sensors is distributed over the area. The readings of these sensors must be transmitted to a base station for further processing. We assume wireless communication and only a limited number of sensors in the range of the base station. Sensors have no location awareness and the network topology is unknown to both sensors and base station. We rely on an underlying communications scheme that forwards messages such that with sufficiently high probability at some point the message reaches the base station.

We were originally inspired by a particular epidemic communication model [4] that uses a randomised swapping scheme in which neighbouring nodes engage in an exchange of messages held in a local cache. The scheme offers a high robustness but at the price of a considerable increase in the amount of data sent. We will not go into further details here since our proposed security provisions do not rely on specific properties of the underlying communication model and have relevance beyond the limited context of epidemic communication.

We do not place any unreasonable restrictions on the attacker. In particular, since sensors are deployed in a potentially hostile environment and the limited budget does not allow for tamper proof sensor nodes, we have to assume that the attacker is able to capture one or more sensors and extract key material. In such a scenario the best we can accomplish at the cryptographic level is a forward secure scheme such that a node capture does not compromise the security of prior messages. At another level the situation in which not all sensor readings can be trusted can perhaps be remedied by placing redundant sensors and assuming the number of captured sensors has a sufficiently low upper bound.

In this paper we strive to strike a balance between the resource constraints and the strong security requirements. We propose a scheme that achieves node to base station authentication with weak freshness and message confidentiality, yet has a modest computational complexity, minimal storage requirements, and quite acceptable message expansion. We do assume an invulnerable and reasonably powerful base station[1]. Our scheme was inspired by a scheme [14] that offers forward secure anonymity through key evolution in the context of Radio Frequency Identification. We have modified that scheme such that it can be used for message authentication, freshness, and confidentiality. Because we do not require anonymity of the nodes, the associated scalability problem disappears. We refer the reader to [2] for an extensive discussion of using key evolution to obtain forward security.

We remark that our confinement to (one-way) node to base station communication enables a solution that is economical in terms of the mentioned resource criteria. For example since we do not require in-network verification or processing of messages from other nodes, we do not need an expensive asymmetric scheme. Note also that a simple single shared network key would not provide much protection in the face of node captures and extraction of key material.

The remainder of this paper is organised as follows. Section 2 gives an overview of relevant existing work. Section 3 introduces our security provisions. Section 4 provides proofs for our claims of authenticity, confidentiality, freshness and forward security. Section 5 investigates implementation options. Section 6 analyses achievements and discusses remaining open problems.

[1] Note that we refer to the central trusted authority as *the* base station but of course it could equally well be distributed over multiple physical base stations provided they each satisfy the given requirements (i.e. reasonable resources, invulnerability).

2 Related Work

General sensor network security surveys include [15, 6, 11]. Closest to our work is [16], in particular SNEP, which is the part based around a single block cipher that is used for authentication, confidentiality, and random number generation. The key difference to note is that SNEP does not provide forward security. Although some empirical results [19] suggest that there is a notable performance penalty for using SHA-1 as our basic building block as opposed to RC5, the latter has the disadvantage that it is covered by a patent. Data from [18] indicates that the widely adopted AES block cipher, which would be a natural replacement for RC5, consumes at least 60% more energy per byte than SHA-1.

TinySec [9] is targeted at the security of node to node communication. The ability to detect in an early stage malicious messages and avoid committing resources to deliver these messages to the destination is used as motivation. We recognise the desirability of early detection possibilities, but failure to address replay attacks—which is justifiably omitted because the required maintenance of state for each communications partner is infeasible for memory limited sensor nodes—raises doubts on the usefulness of investing in authentication of node to node communication. Moreover, a key deployment mechanism is assumed of which the mentioned examples are either not robust (i.e. a single captured node compromises the entire security of the system) or require knowledge of the network topology.

We mention some studies that make further assumptions that are incompatible with the scenario we consider. Secure pebblenets [1] relies on a tamper resistance assumption to protect a shared secret key. SEKEN [7] requires the establishment and maintenance of the network topology by the base station. Sophisticated schemes such as those proposed in [17] require a bidirectional communication channel between node and base station and are clearly beyond what we consider feasible. Security provisions often require that nodes share cryptographic keys with each other. Key establishment mechanisms in sensor networks are considered in [21, 12, 8, 3].

3 Specification

In our discussion we ignore the specifics of the underlying communication model. We use a very general message format as our starting point and define extensions that achieve node to base station authentication, confidentiality and weak freshness. We delegate the actual message sending and forwarding to the underlying communication model.

The constructions we suggest for authentication and confidentiality are a straightforward application of the well known symmetric key scheme. Our main contribution lies in the introduction of a continuously evolving key that achieves forward security and, additionally, provides implicit freshness without the need to expand messages with e.g. counter values.

3.1 Definitions

Let N be the set of nodes in the network. Node $n \in N$ generates messages of the form (n, d) where d is an element of some unspecified data domain.

We assume the possibility of secure node initialisation, that is, the base station is able to securely distribute an initial key to the sensor nodes. More precisely, for each node $n \in N$ the base station chooses an initial key x_n^0 uniformly at random: $x_n^0 \in_R \{0,1\}^k$ for security parameter k and sends x_n^0 to node n such that x_n^0 is a shared secret between the base station and node n. As explained below, this key will be updated after each transmission of node n yielding a series of keys $x_n^0, x_n^1, x_n^2, \ldots$.

Let $\mathcal{H}$ be a non-invertible collision resistant hash function with domain restricted to k-bit strings: $\mathcal{H} : \{0,1\}^k \to \{0,1\}^k$.

We define $h : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$ to be a MAC function that on input of a k-bit key x and arbitrary d outputs a t-bit signature $h_x(d)$ on d. We require h to resist adaptive chosen-message attacks.

3.2 Authentication

We redefine the i-th message of node n to include a signature under ephemeral key x_n^i:

$$(n, d, h_{x_n^i}(d)) \tag{1}$$

Informally we argue that under the assumption that x_n^0 is unknown to the attacker and the fact that h offers key non-recoverability, an attacker is unable to learn any key $x_n^j, j \geq 0$ without physically tampering with the sensor node. Therefore an attacker will have a negligible probability of success in creating a valid signature for data other than data previously signed by n. We provide a more thorough security analysis in Section 4.

3.3 Key Evolution

We define for any $n \in N$ the i-th key x_n^i to be simply the image under $\mathcal{H}$ of the previous key x_n^{i-1}:

$$x_n^i = \mathcal{H}(x_n^{i-1}),\ i > 0 \tag{2}$$

We assume nodes erase key x_n^{i-1} once the new key x_n^i has been determined and the erasure is such that we can assume that physical inspection cannot reveal erased keys.

3.4 Confidentiality

For confidentiality we introduce encryption function $E : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ that on input of a k-bit key x and arbitrary plain text d of length l outputs cipher text $E_x(d)$ also of length l. In our scenario it is particularly important that encryption does not result in data expansion as energy costs are dominated by transmission rather than computation [9, 16, 5]. For the moment we require E to resist adaptive chosen plain text attacks, implementation options and specific security properties will be explored in Section 5.

Messages that offer confidentiality in addition to authentication are now defined by:

$$(n, E_{x_n^i}(d), h_{x_n^i}(E_{x_n^i}(d))) \tag{3}$$

3.5 Freshness

Since our scenario does not exclude the possibility of message loss or delay in message delivery, the base station should anticipate non-sequential message receipt. This is typical behaviour in the epidemic communication model from [4].

We introduce an acceptance window of length $2w+1, w \geq 0$. Let X_n be the element of the hash chain $x_n^i, i \geq 0$ stored by the base station. Initially X_n is set to x_n^0. Upon receipt of a message (n, d, s) with $n \in N$ the base station searches for a $j, 0 \leq j \leq 2w$ such that $h_{\mathcal{H}^j(X_n)}(d) = s$. If it finds such a j it accepts d as authentic and fresh originating from node n.

The stored element of the hash chain is updated as follows.

$$X_n := \mathcal{H}^{j-w}(X_n) \quad \text{iff} \quad j > w \tag{4}$$

Hence apart from a short initialisation phase where the backward window is empty, $\mathcal{H}^w(X_n)$ is the maximal element of the chain $x_n^i, i \geq 0$ for which a message has been received. Once a message signed with an element further down the chain is received, the acceptance window is shifted forwards such that messages with an offset of at most $\pm w$ are accepted as fresh. Obviously the non-invertibility of $\mathcal{H}$ prevents us from storing $\mathcal{H}^j(X_n)$ rather than $\mathcal{H}^{j-w}(X_n)$.

Of course there is a trade-off to be made here, the base station can reserve $2wk$ additional bits per node to avoid having to compute up to $2w$ elements of the hash chain for each received message (n, d, s).

We will ignore the handling of duplicate messages here. It is trivial to record for which elements of the hash chain a message has been received such that duplicates can be ignored if so desired.

We remark that there is ample opportunity for refinement of the base station's policy for accepting messages. The base station could, for instance, dynamically change the value of w perhaps even for each node individually. It could use previous communication patterns and the current network traffic to determine an appropriate value. Should the base station detect prolonged disruption of communication it could temporarily increase the length of the forward window or update the stored hash chain elements using previous patterns to avoid loss of synchronisation. These refinements are context dependent and will further be ignored here.

4 Verification

4.1 Signature Forgery

We define a signature forgery attack to be successful if the attacker without knowledge of key x succeeds in constructing some message (n, d, s) that is accepted by the base station as authentic and fresh originating from node n using key x, when no such signature under key x has been generated by node n.

We show that the probability of success of such a forgery is negligible in security parameters k and t. In our analysis we consider h to be a black box for which the probability of constructing a valid signature $s = h_x(d)$ for arbitrary data d and unknown but fixed key x is at most 2^{-t}, assuming $t \leq k$.

Note that for specific node $n \in N$ the set of acceptable keys is determined by $\{\mathcal{H}^i(X_n) \,|\, 0 \leq i \leq 2w\}$. It now follows easily that the probability of successful forgery is at most $2^{-t+\log(2w+1)}$.

As expected, the width of the acceptance window influences the probability of successful forgery. When determining an appropriate value for t also an upper bound on the value of w should be established such that the probability of successful forgery as determined above is sufficiently low. We refer to Section 5 for practical values for both t and w.

A notable difference with [14] is that an attacker is forced to choose a particular node $n \in N$ whereas [14] excludes node identifiers in order to preserve anonymity. This increases the probability of successful forgery by a factor $|N|$.

4.2 Confidentiality Violation

We define oracle $\mathcal{O}_C$ that on input of arbitrary plaintext d, node $n \in N$, and iteration number $i \geq 0$ outputs the cipher text under key x_n^i:

$$\mathcal{O}_C(d, n, i) = E_{x_n^i}(d) \tag{5}$$

We consider an attacker that has queried $\mathcal{O}_C$ at a number of inputs (d_j, n_j, i_j) that is polynomial in security parameter k but not at input (d, n, i). We define an attack on the confidentiality of $E_{x_n^i}(d)$ to be successful if the attacker is able to determine any information on d other than its length $|d|$.

First, we show by contradiction that the key that is input to E is unique. Assume $x_n^i = x_{n'}^j$ with $i < j$. Note that we do not require the keys to be element of the same hash chain. Then we have established x_n^{i-1} and $x_{n'}^{j-1}$ as colliding inputs to $\mathcal{H}$, contradicting its collision resistance.

If we consider E to be a Random Oracle and the input to E is unique, we have that $E_{x_n^i}(d)$ cannot be distinguished from a truly random $r \in_R \{0,1\}^{|d|}$ without knowledge of key x_n^i. Hence the probability distribution is uniform over all possible plaintexts from $\{0,1\}^{|d|}$.

4.3 Forward Security

Suppose an attacker is able to physically extract or otherwise obtain at some point key x_n^j. We argue that knowledge of this key does not lead to a compromise of any previous messages.

We define oracle $\mathcal{O}_F$ that provides for given $n \in N$ and $i \geq 0$ key x_n^i:

$$\mathcal{O}_F(n, i) = x_n^i \tag{6}$$

We consider an attacker that has queried $\mathcal{O}_F$ at input (n, j) but not at any of the inputs (n, i), $0 \leq i < j$. We show that knowledge of the obtained key x_n^j does not help the attacker in forging signatures or deciphering messages under any key $x_n^i, 0 \leq i < j$.

Under the assumed non-invertibility of $\mathcal{H}$ key $x_n^{j-1} = \mathcal{H}^{-1}(x_n^j)$ cannot be determined from x_n^j in less than 2^k operations. Obviously we then have for any $x_n^i, 0 \leq i < j$ that it cannot be determined from x_n^j in less than 2^k operations since any such x_n^i would lead to $x_n^{j-1} = \mathcal{H}^{j-1-i}(x_n^i)$.

5 Implementation

So far our discussion has been on an abstract level and mainly focused on the security properties. In order to analyse the resource requirements we will now consider implementation options for abstract functions $\mathcal{H}$, h, and E as defined earlier.

5.1 Hash Function

Although recently it has been shown [20] that finding collisions for SHA-1 requires far less effort than the ideal strength of 2^{80} operations, we propose to take SHA-1 to implement $\mathcal{H}$. We note that the non-invertibility of SHA-1 is not affected and we have only used its collision resistance property in support of our confidentiality proof of Section 4.2. Even there, the inputs to $\mathcal{H}$ cannot be influenced by the attacker which makes the attack inapplicable.

The energy required to establish a new key as in equation (2) is about 15 μJ according to experimental results from [18] and assuming $k = 160$. The code size for a software implementation on various embedded processors is around 2000 byte based on data from [19] which is less than half the amount required for MD5. Presumably, in a hardware implementation several optimisations can be applied to reduce the required chip area, however the SHA-1 implementation is likely to be significant compared to the simple sensor node logic required for e.g. a smoke detecting sensor. It is our objective to reuse SHA-1 for functions h and E in order not to further increase code size/chip area.

5.2 MAC Function

We choose to use HMAC-SHA1-t as our implementation of MAC function h. Here parameter t is used as described in [10] to denote that only the first t output bits are used as the MAC.

We remark that the HMAC construction involves basically two applications of the hash function and some simple padding operations. The added complexity in terms of code size/chip area is therefore minimal. The amount of energy required to generate a signature is estimated by [18] to be around 1 μJ/byte.

5.3 Encryption Function

Similar to [14] we define $\mathcal{G}$ to be a non-invertible collision resistant hash function independent from $\mathcal{H}$ with domain restricted to k-bit strings: $\mathcal{G} : \{0,1\}^k \rightarrow \{0,1\}^k$. We use the hash values generated by $\mathcal{G}$ as a keystream. E can now be defined in terms of $\mathcal{G}$:

$$E_{x_n^i}(d) = d \oplus \mathcal{G}(x_n^i) \tag{7}$$

Where $\oplus$ denotes the bitwise exclusive-or operation that is understood to discard the remaining suffix of $\mathcal{G}(x_n^i)$ of length $k - |d|$ bits and operate on the first $|d|$ bits of $\mathcal{G}(x_n^i)$ only.

As a first observation note that the length of the plain text is now limited to k bits. We remark that for the scenarios we consider the value of k (to be determined shortly) is more than sufficient to accommodate typical message sizes. We could generate a longer keystream, for instance, by applying $\mathcal{G}$ to simple variations of x_n^i.

Earlier we showed uniqueness of key x_n^i. If we consider $\mathcal{G}$ to be a Random Oracle, we can view $\mathcal{G}(x_n^i)$ as a One Time Pad offering perfect secrecy. Non-invertibility of $\mathcal{G}$ also guarantees that a known plaintext attack in which an attacker obtains $\mathcal{G}(x_n^i)$ does not lead to a compromise of x_n^i.

Notice that the construction we suggest here relies rather heavily on the assumptions that we make on the properties of $\mathcal{G}$. Once we choose a concrete hash function to implement $\mathcal{G}$ that has some predictable bias, this bias would directly reveal (partial) information on the plain text.

An attacker knowing some plaintext can determine (a prefix of) keystream $\mathcal{G}(x_n^i)$ resulting in the compromise of subsequent keys if $\mathcal{G} = \mathcal{H}$. However from the point of chip area/code size it would be desirable if $\mathcal{G}$ and $\mathcal{H}$ could use the same concrete hash function. Therefore, we define $\mathcal{G}(x_n^i) = \mathcal{H}(x_n^i \oplus msk)$ for fixed and public $msk \neq 0$ of length k bits. Knowledge of $\mathcal{G}(x_n^i)$ does not allow an attacker to determine subsequent key $\mathcal{H}(x_n^i)$.

5.4 Choice of Security Parameter Sizes

Throughout we have used k to denote the size of the key and t to denote the size of the signature. As mentioned before the amount of message expansion is of particular importance in our scenario.

We adopt a keylength of $k = 160$ bit as suggested for a collision resistant hash function in [13]. Choosing a lower value does not seem to offer significant savings as the key is expanded to full block length when used as input to SHA-1.

The suggested lowerbound of $t = 64$ bit for the size of the MAC could add significant overhead for typical message sizes. As mentioned in [13] for specific applications this value could be reduced if the chance of successful forgery $2^{-t+\log(2w+1)}$ is acceptable, taking into account the total number of forgery attempts an attacker is able to perform during the lifetime of the sensor network and the implications of a single successful forgery.

6 Conclusions

We developed a set of security provisions for communication in wireless sensor networks which establishes authentication of the origin of data, confidentiality of data, forward security (implying a weak form of tamper resistance), and freshness (to mitigate the effect of maliciously delayed data).

Our security provisions do not protect against attacks at the physical level where an attacker directly manipulates sensors, for instance, by placing a heat source in close proximity of a temperature sensor or shielding a sensor from its environment. We are also not concerned with availability and do not prevent the attacker to learn information from traffic analysis.

We strove to minimise the resource requirements for the sensor nodes. The computational complexity is low due to our choice of using a hash function. In order to minimise the communication overhead, we chose for encryption algorithms without data expansion. Finally, by reusing already available logic (the hash function) chip area can be reduced.

Rather than a communication protocol, we developed a set of security provisions that can be superimposed on several underlying communication models for sensor networks. This made it possible to broaden the scope of our work which was initially targeted at adding security provisions to the epidemic model as described in [4].

The decision to secure the data transfer at the level of node to base station communication has several consequences. It allowed us to minimise the resource requirements for the nodes and to look for solutions without PKI or keys shared between nodes. Of course this comes at the price of an increased effort at the base station. As a drawback, we have that messages will only be verified at the base station and not during their transmission through the network. This could make a denial of service attack by inserting false messages more effective. In order to assure freshness, we have introduced an acceptance window at the base station. A drawback is that this will imply that in some cases benign messages will be ignored. In practice the size of the window will have to be tuned to the actual latency of the network.

There are several interesting options for future research. First of all, a simple protection against DoS attacks would be useful to strengthen our scheme. Next, practical experiments are needed to validate that the degradation of the performance of the epidemic communication model stays within reasonable bounds when adding the security provisions. Finally, it would be interesting to study secure bi-directional communication between the nodes and the base station, making it possible to dynamically instruct the sensors.

Acknowledgements. Thanks are due to Berry Schoenmakers for his helpful comments on the verification and to Maarten van Steen for introducing us to the Gossip communication model.

References

1. Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenets. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 156–163, New York, NY, USA, 2001. ACM Press.
2. Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. Cryptology ePrint Archive, Report 2001/035, 2001.
3. Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM Press.
4. Daniela Gavidia, Spyros Voulgaris, and Maarten van Steen. Epidemic-style monitoring in large-scale sensor networks. Technical Report IR-CS-012.05, Vrije Universiteit Amsterdam, March 2005. `http://www.cs.vu.nl/pub/papers/globe/IR-CS-012.05.pdf`.
5. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.
6. F. Hu and N. Sharma. Security considerations in ad hoc sensor networks. *Ad Hoc Networks*, 3(1):69–89, 2005.
7. Kamran Jamshaid and Loren Schwiebert. Seken (secure and efficient key exchange for sensor networks). In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 415–422, 2004.
8. K. Jones, A. Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy. Towards a new paradigm for securing wireless sensor networks. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 115–121, New York, NY, USA, 2003. ACM Press.
9. Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM Press.
10. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication, February 1997. RFC 2104.
11. Y. W. Law, S. Dulman, S. Etalle, and P. Havinga. Assessing security-critical energy-efficient sensor networks. Technical Report TR-CTIT-02-18, University of Twente, The Netherlands, July 2002. http://purl.org/utwente//38381.
12. Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.
13. Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.
14. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.
15. Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.
16. Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

17. Asad Amir Pirzada and Chris McDonald. Kerberos assisted authentication in mobile ad-hoc networks. In *CRPIT '04: Proceedings of the 27th conference on Australasian computer science*, pages 41–46, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
18. Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. Analyzing the energy consumption of security protocols. In *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35, New York, NY, USA, 2003. ACM Press.
19. Ramnath Venugopalan, Prasanth Ganesan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Encryption overhead in embedded systems and sensor network nodes: modeling and analysis. In *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 188–197, New York, NY, USA, 2003. ACM Press.
20. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. Technical report, Shandong University, Shandong, China, June 2005.
21. Yongge Wang. Robust key establishment in sensor networks. *SIGMOD Rec.*, 33(1):14–19, 2004.

Low Rate DoS Attack to Monoprocess Servers

Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo, and Pedro García-Teodoro

Dpt. of Signal Theory, Telematics and Communications - University of Granada,
c/ Daniel Saucedo Aranda, s/n - 18071 - Granada, Spain
{gmacia, jedv, pgteodor}@ugr.es

Abstract. In this work[1], we present a vulnerability in monoprocess or monothreaded servers that allows the execution of DoS attacks with the peculiarity that they are generated by low rate traffic. This feature makes the attack less vulnerable to detection by current IDS systems, which usually expect high rate traffic. The intruder can take advantage of some knowledge about the inter-output times in the server to build the attack. We have simulated and tested it in a real environment, obtaining worrying conclusions due to the efficiency achieved by the attack, with low effort from the attacker.

1 Introduction

Denial of Service (DoS) attacks are a very serious problem in Internet, with a great impact in the service offered by small as well as by big and important companies like Ebay, Amazon or Buy.com [1]. Besides, this kind of attacks could be carried out in a wide variety of manners, as it has been pointed out in the computer network literature [2]. This kind of attacks tries to exhaust some resources in a single machine or in a network with the aim of either reducing or subverting the service provided by them. Considering the three main aspects of security, that is, confidentiality, integrity and availability, DoS attacks focus their target in the reduction of the last one. The intruders usually achieve their goal by sending to a victim, either in a single or in a distributed way (DDoS) [2], a stream of packets that exhaust its network bandwidth or connectivity, or somehow exploiting any discovered vulnerability, causing all the cases a denial in the access to the regular clients. In recent years, there have been some large-scale attacks that affected important Internet sites [3] [4] [5], what demonstrate the vulnerability of the network environments and services to this kind of attacks.

Close to the evolution of the DoS attacks, many proposals have also appeared for preventing and detecting them. Many of the preventive measures are applicable to mitigate DoS attacks, like egress or ingress filtering [6] [7], disabling of unused services [8], change of IP address, disabling of IP broadcasts, load balancing, or honeypots [9]. However, although prevention approaches offer increased security, they can never completely remove the threat so that the systems are always vulnerable to new attacks.

[1] This work has been partially supported by the Spanish Government through MYCT (Project TSI2005-08145-C02-02, FEDER funds 70%).

That is why it is necessary the adoption of intrusion detection systems (IDS) capable for detection of attacks [10]. Many efforts have been made to solve the problem of discovering DoS attacks in a high-bandwidth aggregated traffic, as it is stated in the works of Talpade et al. [11], Cabrera et al. [12] or Mirkovic et al. [13]. Besides the signature detection IDS's, the major part of the proposed IDS's for DoS detection relies on the identification of the attack by using techniques that are based on the hypothesis that it is going to be carried out through a high rate flooding from the intruder.

In this paper we analyze an exploit in monoprocess or monothreaded servers. We postulate the use of these kind of servers in simple device services in pervasive computing, so this exploit allows an intruder to carry out a kind of DoS attack, classified as an application level attack according to [14], that presents the special feature of using a low rate traffic against the server. Due to this fact, the attack will be capable of bypassing the detection mechanisms that usually rely on a high-bandwidth traffic analysis. Kuzmanovic et al. already presented in [15] an attack with similar rate characteristics and, afterwards, some solutions appeared to tackle this problem [16], [17], [18]. Although the attack presented in this work is similar to the previous ones in some aspects, there are important differences with them. Both type of attacks take advantage of a vulnerability caused by the knowledge of a specific time value in the functioning of the protocol or application, thus allowing an ON/OFF waveform attack that results in low rate traffic but with high efficiency in denying the target service. However, the attack presented in [15] is TCP-targeted, while the one introduced here threatens the application level. While the other attack generates outages in a link, this one only overflows a service running in a server, so that it could be unnoticeable in a link overflow monitoring. Moreover, in our case, the waveform of the attack is changeable along the time and the burst period does not involve a very high rate, so that the previously presented detection techniques require to be adapted. There are also differences in the vulnerabilities exploited in both kind of attacks. In the TCP-targeted (low rate) case, the knowledge of the RTO timer for the congestion control implemented in TCP is exploited, whilst in the monothreaded server case the inter-output times are the key to build the attack, as it will be presented in Section 3. But the main difference between the two attacks lies in the fact that during the TCP-targeted attack, the link is just busy in the outages periods, while in the ours one the server is always busy and processing requests from the intruder, causing the legal users the sensation that the server is not reachable. This last feature is similar to the behavior of the `Naptha` attack [19], although the main difference is that `Naptha` is a brute-force attack executed with high rate traffic.

This article is structured as follows. In Section 2 the scenario of the attack and the model of the system that is going to be attacked are defined. Next, in Section 3 it is specified how the attack is going to be performed. Section 4 will evaluate the effectiveness and behavior of the attack by means of simulations. Following, Section 5 will validate the results in a real environment, showing the goodness of the simulated results. Finally, some conclusions are presented in Section 6.

2 Modelling and Scenario Definition

We shall analyze an exploit in monoprocess or monothreaded servers that would potentially allows an intruder to carry out denial of service attacks at the application level characterized by low arrival rates at the server. The scenario to be considered is a generic client-server scenario where the server is going to receive aggregated traffic from legal users and intruders. Without lost of generality, the intruder could make the decision of carrying out the attack in a distributed manner or not, depending on the preventive measures [14] implemented in the target system and the own capabilities of the attacker.

2.1 Server Modelling

The first step in this work is to define a model that captures the behavior of a monoprocess or monothreaded server. The server will be simply a black box that receives packets, processes them and gives responses. This way, the overall system receives input messages requesting some specific service offered by a server. The system reacts to these requests by sending the message carrying the desired data (OK message in advance), or simply rising an event or sending a message indicating an overflow in the internal buffers or queues (MO event in advance). It is important to point out that, in real environments and depending on features of the considered server, this last event could be observable either through a specific protocol message, or logs, process signals, or even to be not observable at all.

The model consists of two elements: a) a *service queue* and b) a *service module*. These elements interact as shown in Fig. 1, where the arrows show the processing path followed by the incoming packets. The first building block is the service queue, where the incoming requests are temporarily stored. It could represent a TCP connections buffer, UDP buffer or simply internal application buffers. The processing of the client requests is made by the service module, which represents a service carried out by a single processing thread running on an operating system. That is, this module serves only one request from the service queue each time (iterative server).

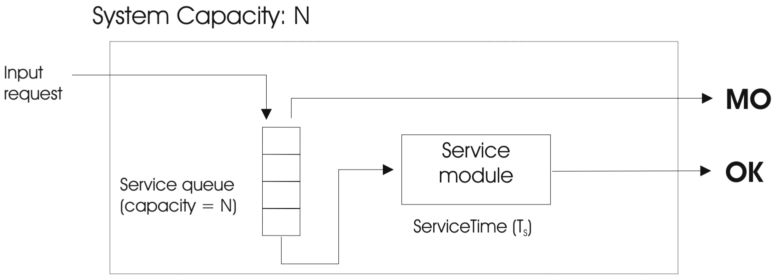


Fig. 1. Model for monoprocess/monothreaded server

The system operates as follows. An input request enters the system. If the service queue has free positions, the request is queued on it. Otherwise, an MO event occurs. The request will stay in the service queue during t_q seconds waiting for its turn to be served. This time is the so called *queue time* in queuing theory. Then, it will be processed by the service module during t_s seconds. This time is called *service time* and it corresponds to a parsing of the data, or simply the time elapsed in a complex calculation. Finally, when the processing is completed, the corresponding response to the input request is generated (OK message).

We can consider that t_q and t_s are samples from respective random processes represented by random variables. As a notation, in the following we will use lower-case letters for instantaneous values of a random variable, and their correspondent capital letters represent the general process. Moreover, operators $E[\cdot]$ and $Var[\cdot]$ will be, respectively, the mean value and the variance of the specified random variable. The variable T_s will be modelled as a normal distribution ($\mathcal{N}$).

The arrival of the user requests is also considered a random process. As recommended in many teletraffic studies [20], the arrivals are modelled by a Poisson process. Therefore, the distribution of user inter-arrival times, T_a, corresponds to an exponential probability:

$$P(T_a = t) = \lambda \cdot e^{-\lambda t} \tag{1}$$

where λ represents the arrival rate of requests from the users. Moreover, it is demonstrated [21] that the aggregation of traffic from users whose inter-arrival times are exponentially distributed, with rates λ_i, results in a traffic with inter-arrival time that also follows an exponential distribution with inter-arrival time:

$$\lambda = \sum_{i=1}^{n} \lambda_i \tag{2}$$

This expression implies that it is possible to suppose only one user in the scenario of study to represent all the aggregated traffics (Poisson processes) coming from n users.

2.2 Inter-output Time

The potential exploit that allows an intruder to carry out an attack is the knowledge of the inter-output time of the server. This parameter, named τ, is defined as the time elapsed between two consecutive outputs or answers given by the server. As stated in Section 2.1, an OK message is here termed an output or answer. On the contrary, MO messages are not going to be considered as outputs.

For simplicity, we examine a case of study in which fixed times are considered instead of random processes. Thus, the service time will be represented by the mean value of its random variable. Although this could seem a very restrictive case, it will be shown in the next section that the results are still valid for more complex cases in which the service time is not fixed.

With the above restrictions and assuming that the service queue is always full of service requests, Fig. 2 shows an example of the inter-output time diagram

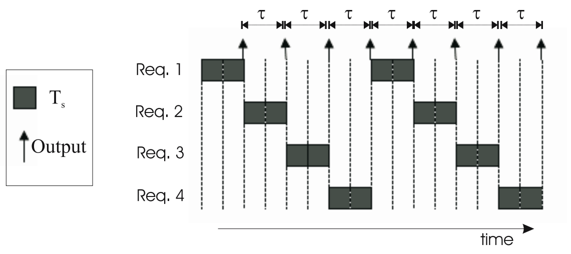


Fig. 2. Inter-output time diagram and processing state for several requests

and processing state for several requests. For a server with a service queue of 4 positions, the service time is drawn in dark grey colour. Outputs are signaled by vertical arrows. A simple observation of the scheme leads us to the conclusion that, upon the condition of fixed service time, it is obtained a fixed inter-output time τ that corresponds with:

$$\tau = E[T_s] \tag{3}$$

In the case that the service queue is not always full of service requests, the inter-output time could experience some variations, specifically when there are no requests to be served. However, the attacker can easily maintain an appropriate level of requests in the queue in order to get a fixed inter-output time.

The low rate DoS attack to monoprocess servers relies on the knowledge of this time. Nevertheless, it is necessary to indicate that this time could be still estimated by the intruder in real conditions where there are variable service times and propagation delays. In the case that variable service times and propagation delays appear, it is important to differentiate between the inter-output time from the server and that one observed by the users (legal or not). In effect, an observer of the inter-output time located at the server will appreciate a normal distribution behaviour with a mean $E[T_s]$ and a variance $Var[T_s]$:

$$\tau_{server} = \mathcal{N}\big(E[T_s], Var[T_s]\big) \tag{4}$$

However, any client of the service will appreciate a difference in the inter-output time, due to the influence of the variance in the round trip time (RTT). This way, assuming independency between the variables T_s and RTT, the observation of the inter-output time will be, in this case:

$$\tau_{user} = \mathcal{N}\big(E[T_s], Var[T_s] + Var[RTT]\big) \tag{5}$$

In what follows, we will try to validate this behaviour by means of simulation.

2.3 Inter-output Time Validation

We have used Network Simulator (NS2) [22] to check whether the assumption that inter-output time approximates to Eq. (5) is correct under non empty queue conditions. For this purpose, a new class derived from the application class that behaves like a server (*server class*) has been implemented, which makes traces and logs to track all the events and statistics necessary for our study.

We have used two set of experiments to validate the expected results about the inter-output time of the model. The first set of experiments tries to validate the behaviour explained in Section 2.2, when fixed values for T_s are used. According to the results obtained in this case, it can be concluded that the behaviour of the simulated system is as expected.

In the second set of simulations, we have considered the case when the service time T_s and the round trip time RTT are modelled with normal distributions. The obtained results are very promising, showing low variations, in a statistical sense, among the predicted and the simulated behaviour.

Fig. 3 shows some partial results derived from the second set of experiments. In this case, the service queue has been offered user traffic with $E[T_a] = 1.0$ s. Other values used in this simulation are the capacity of the queues $N = 40$, and $T_s = \mathcal{N}(1.5\,s, 0.02)$. The round trip time takes the value $RTT = \mathcal{N}(0.6\,s, 0.02)$. It should be notice that $E[T_a]$ has been adjusted to maintain the service queue completely full, due to the fact that the user traffic rate is higher than the service rate in the server. The simulation for a period of 1000 s provides 694 outputs. The mean value obtained for inter-output time is 1.520 s, with variance 0.041, which accurately approximates the values given by Eq. (5). It is noticeable that the obtained histogram for the inter-output times approximates to the normal probability function (see Fig. 3(b)).

Fig. 4 shows other results belonging also to the second set of experiments, where there is no congestion in the service queue. In this simulation case we have adjusted the same parameters than in the previous one, but changing the user inter-arrival mean time $E[T_a]$ to 1.5 s. The same value for $E[T_s]$ have been chosen in order to maintain requests in the service queue. The simulation for a period of 1400 seconds has provided 923 outputs. The mean value obtained is 1.536 s and the variance 0.114. The deviation is greater than in congestion conditions due to the values generated when the buffer has no requests. Fig. 4 represents the inter-output times in the main axis and the occupation of the buffer in the secondary axis (dashed lines). We can see that the periods with void service queue occupation results in sporadic higher values for the inter-output times, as stated in Section 2.2.

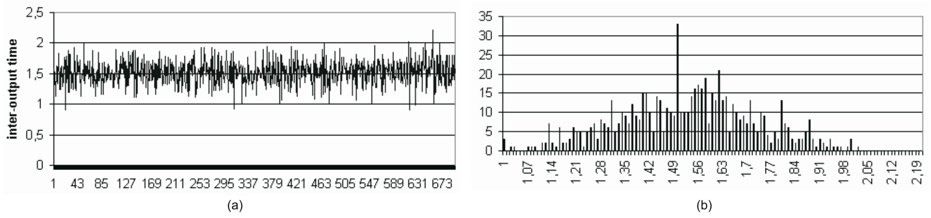


Fig. 3. Simulation of inter-output time with flooded buffer: (a) Inter-output time values and (b) histogram of the samples

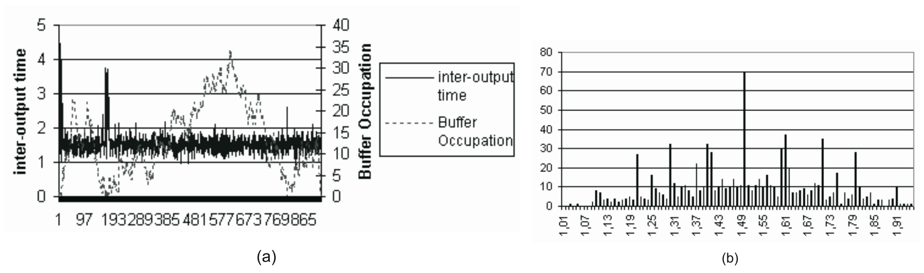


Fig. 4. Simulation of inter-output time with no flooded buffer: (a) Inter-output time values and buffer occupation level and (b) histogram of the samples

The results of this set of experiments show that the assumptions made for the operation under congestion in the service queue are good enough. It is also interesting to check that, under no complete saturation conditions, the results are still valid although they present less accuracy.

These experiments show that, even in simulated scenarios where service time is variable, the inter-output time could be still predictable for a possible intruder to build an attack based on this knowledge.

3 Low Rate Attack Specification

With the aim of demonstrating that monoprocess applications could suffer a low rate denial of service attack, we are going to specify how the attack would be accomplished.

In order to successfully attack the server, first of all, it is necessary to determine the weak point to be exploited. The resource to be exhausted is the service queue, in which the requests are queued by the application. Therefore, the objective is to maintain it full (or at least so much full as possible) of requests from the intruder. This way, legal users are not going to be able to queue their requests, thus perceiving a denial of the service provided by the application.

This strategy is commonly used in DoS attacks. However, the particular characteristic of the introduced attack here is that the intruder is not going to flood the buffer with a high rate of requests (brute-force attack), but a low rate traffic. The way to achieve this goal is by making a prediction of the instants at which the server responds to the requests, that is, when an output is going to be generated. The intruder will flood the service queue only during a short period of time around the predicted output time, resulting in an overall low rate flood experienced by the target server.

The low rate attack will have two phases: a *transitory phase*, detailed in Section 3.2, which will pursuit to initially flood the service queue, and a *permanent phase*, explained in Section 3.1, through which we try to maintain it full (or at least so much full as possible) of requests by sending request packets at a low rate.

3.1 Permanent Phase Execution

The aim of the permanent phase of the attack is to maximize the time during which the service queue is full of requests. This implies that every released position in the queue has to be seized by an intruder request in the minimum time. When the system is full of requests, a free position is issued when the server provides an output or answer. It is therefore necessary to concentrate the intruder efforts in the moment in which the server raises it, in order to seize the new free position in the service queue. Therefore, the attacker should, ideally, send the requests in such a way that they arrives at the server synchronized with the outputs.

We concluded in Section 2.2 that the perceived time between two consecutive outputs is a random process represented by τ_{user}. However, the statistical nature of the process will introduce some variability in the instantaneous values of τ_{user}, which should be considered in the attack strategy. It is also important to note that the inter-output time is independent of the queue time, and only the service and round trip times determine it. Besides, at the time we are trying to seize positions in the service queue, there will be other users that are attempting to succeed on this same purpose. So, in order to raise the probability of filling the free position with low rate traffic when an output happens, the intruder should synchronize the sending of the requests with the output in the server. To do this, the intruder use a ON/OFF scheme to flood the server with the ON phase situated in the moment of the output.

The proposed attack strategy consists in consecutive periods composed by a period of activity, *ontime*, followed by a period of inactivity, *offtime*. The attack is specified by the following parameters, as depicted in Fig. 5:

– *Estimated mean inter-output time* ($\overline{E[\tau_{user}]}$): it is an estimation of τ_{user} made by the intruder. This can be easily obtained by sending close requests and evaluating the time between the corresponding responses. Of course, the answers should be consecutive in the server. After some proofs, the intruder will succeed on this aim.
– *Offtime:* time during which there is no transmission of attack packets.
– *Ontime*: time during which an attempt to flood the service queue is made by emitting request packets.

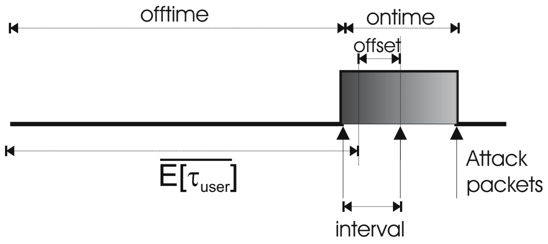


Fig. 5. Attack parameters

– *Interval*: period of time elapsed between the sending of two consecutive packets during *ontime*.
– *Offset:* represents the deviation between the center of *ontime* and $\overline{E[\tau_{user}]}$. The value 0 for the *offset* indicates that *ontime* is completely centered in $\overline{E[\tau_{user}]}$.

Both *offtime* and *ontime* must be adjusted in such a way that a synchronization between the output and the reception of the requests from the intruder is achieved. Thus, *ontime* should be centered around the estimated mean inter-output time and its duration should be proportional to twice its variance to account for its variability. The *offset* is considered to shift the whole *ontime* period in order to take into account second order aspects, as those related with discrete timing in the attack packets during *ontime*.

Although the main strategy of the permanent phase has been outlined, there are some important details to be considered. First, the effect of the statistical variability of T_s and RTT should be taking into account. In order to avoid the cumulative sum of variations through several periods, the intruder will reestimate on each period the expected timing for the next output. So, the attack period (*offtime / ontime*) will be restarted each time a response is received from the server. Obviously, the reception of an output during the *ontime* period implies to stop the current period start a new one. Besides, it is important to take into account the effect of the round trip time between the intruder and the victim. To consider this, the initially calculated *offtime* value should be reduced to *offtime*$-E[RTT]$.

Fig. 6 shows a diagram of the execution of the permanent phase that can be used to review the different steps of the attack. First, the *ontime* period starts by sending packet P1. Packet P2 is sent before the output O1 arrives from the

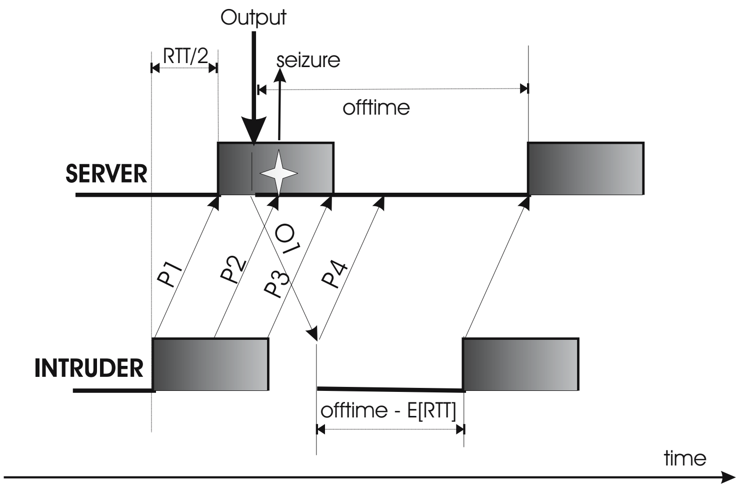


Fig. 6. Permanent phase

server. After that, the packet P2 seizes the new free position appeared after the response (O1). However, the intruder continues the *ontime* period till the reception of the response (packet O1). Meanwhile, the intruder sends the packet P3, which will cause an overflow event in the server upon its reception. On the sending of this last packet during the *ontime* period, the intruder will schedule a timer with value *offtime* and the new period begins. But, on the reception of the answer to packet P1 (packet O1), the period stops and a new one is started.

This mechanism is suitable to achieve our goal. Nevertheless, there is another case to consider. What happens if the output is raised after the reception of the whole ontime period in the server? In this case, the free position would be available till the next *ontime* to be seized by a user. This implies a high probability of failure in the aim of seizing the free position. To avoid this effect, on the reception of an answer from the server, the intruder will send another attack packet (packet P4 in Fig. 6) and reschedule the period with $offtime - E[RTT]$. Obviously, when the intruder does not receive the response (e.g., the output in the considered period corresponds to a request made by a legal user instead of the intruder) and the output is raised after *ontime*, it is more probable to fail the seizure. Thus, we can conclude that seizure failures, after a delay due to the queue time, increase the probability of new failures.

3.2 Transitory Phase Execution

The aim of the transitory phase is to initially flood the service queue. This phase could be made by high rate flooding, although this would make the attack more detectable. However, if the rate of the flooding is slightly higher than the service rate in the server, the queue will end up flooded after some periods. That is, initially the attacker will acquire some positions in the service queue. If the methods associated to the permanent phase are applied, those positions will remain captured, while the excess of traffic would eventually add seized positions from the intruder. This way, the attacker will finally get the whole queue full of his/her requests.

Usually, the intruder will detect the complete flooding of the service queue by means of the observation of the rejection of his/her packets. For example, in a connection oriented application the reception of RESET messages answering the SYN requests (in the case of TCP connections) will identify the complete flooding.

4 Attack Evaluation

In order to check the efficiency of the attack, and to validate the current work, it is convenient to establish some indicators to measure its performance:

- The *percentage of seizures* (S) as the number, in percentage, of buffer positions seized by the attack divided by the total number of seizures experienced in the server, within a specific period of time. This is a measure of the efficacy of the attack.

– The *effort of the attack* (E) as the number, in percentage, of MO packets received by the intruder, divided by the total number of packets generated by the intruder. This value represents the effort made by the attacker.
– The *user success percentage* (U) as the ratio, in percentage, between the number of seizures made by a legal user and the total number of packets generated by him/her. U is a measure of the perception on the service level experienced by a given legal user.
– The *overflow percentage* (O) as the ratio, in percentage, of the total MO messages divided by the number of packets generated by the server. This parameter tells about the saturation experienced by the server.

Among all the indicators, only the effort of the attack is observable by the intruder during the attack. This is because only the server knows the total number of packets and seizures generated during the observation period, which are necessary figures to calculate S and O. On the other side, U will be observable only by the legal users that try to access to the server by usual methods.

The aim of the attack is to minimize U, that is, the user perception of the availability of the server. This is similar to maximize S, the more the server is engaged more time with intruder requests, the more the user success percentage decreases. Besides, in order not to be detected by any active intrusion detection system, the attack should also minimize its effort, E. On the other hand, minimizing E will contribute to a lower overflow O in the server, making thus the attack more undetectable.

In order to evaluate the performance and effectiveness of the attack, we have implemented within Network Simulator (NS2) [22], a new object derived from the application class that behaves like the intruder (*intruder class*) and implements the two phases specified in Section 3.

We are interested in discovering how much effective the low rate DoS attack could be. A set of attacks has been tested in the simulator, and worrying results have been obtained because of the effectiveness constated. Fig. 7 shows the results obtained from an example of one attack simulation composed by 1332 outputs. The attack has been launched against a server with $T_s = \mathcal{N}(3.0\,s, 0.2)$ and $E[T_a] = 4.0\ s$. The parameters of the attack have been tuned choosing values for *ontime* = 0.6 s, *offtime* = 2.7 s, *offset* = 0 s, and *interval* = 0.3 s.

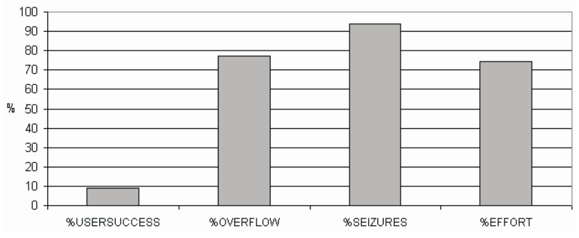


Fig. 7. Effectiveness for one attack

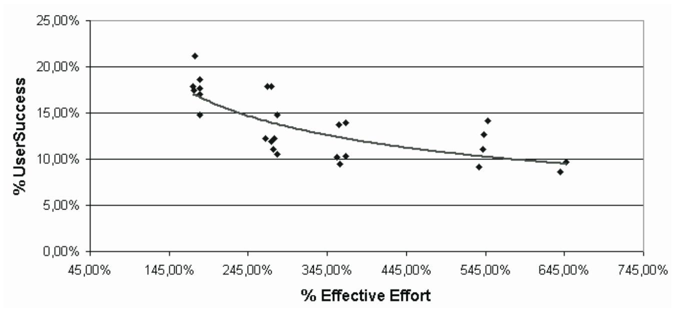


Fig. 8. User success and effective effort for 25 different configurations of the low-rate DoS attack

Additionally, the round trip time has been set to $RTT = \mathcal{N}(0.6\,s, 0.2)$ and the capacity of the system is $N = 20$.

It can be observed that a very high efficiency is obtained, given by $S = 94\%$ and $U = 9.04\%$. On the other side, the value of the *overflow percentage*, $O = 77.08\%$, indicates that the traffic offered to the server by both the users and the intruder is four times its capacity. If the intruder wanted to bypass an IDS system able to detect attacks on this rate, effectiveness should be sacrificed to reduce the overflow rate. This could be easily achieved by reducing *ontime* or increasing *interval*. In order to know the amount of traffic that the intruder generates, the *effective effort* is defined as the traffic offered by the intruder normalized by the capacity of the server:

$$\text{effective effort} = \frac{\text{packets sent by intruder}}{\text{total packets accepted by server}} \tag{6}$$

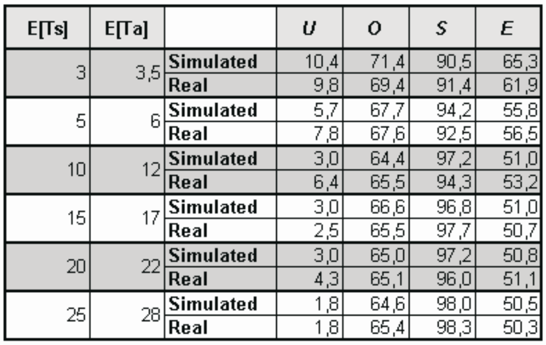
In Fig. 8, 25 possible attacks to the previously defined server are shown. The user success percentage and the effective effort are represented in main and secondary axes, respectively. It can be seen that there are a lot of possible configurations eligible for the attack in order to bypass IDS systems and, at the same time, obtain the best efficiency. As it has already been deduced in previous sections, a better efficiency implies a higher effort, as it can be seen in the tendency line.

Although these values could indicate a minor effectiveness, the results give us a good idea about the performance of the attack.

5 Real Environment Validation

We have tested also the proposed attack in a controlled real application to check its validity. The selected server is a web server that keeps the condition of serving requests in a monothreaded way. Although this is not a typical monothreaded application, it can be configured this way for this purpose. Former extension of the attack to multithreaded servers has motivated the election

Table 1. Real time and simulation values for selected experiments



E[Ts]	E[Ta]		U	O	S	E
3	3,5	Simulated	10,4	71,4	90,5	65,3
		Real	9,8	69,4	91,4	61,9
5	6	Simulated	5,7	67,7	94,2	55,8
		Real	7,8	67,6	92,5	56,5
10	12	Simulated	3,0	64,4	97,2	51,0
		Real	6,4	65,5	94,3	53,2
15	17	Simulated	3,0	66,6	96,8	51,0
		Real	2,5	65,5	97,7	50,7
20	22	Simulated	3,0	65,0	97,2	50,8
		Real	4,3	65,1	96,0	51,1
25	28	Simulated	1,8	64,6	98,0	50,5
		Real	1,8	65,4	98,3	50,3

of this real environment. Therefore, Apache 2.0.52, configured with the directive `ThreadsPerChild = 1`, presents a monoprocess behaviour. This way, we ensure that the processing of a request blocks those awaiting in the service queue. Also, we have considered that every request consists in a connection request. The attack establishes connections and sends no messages on them, letting the web server to close the connection after a timeout period specified by the Apache directive `Timeout`. This directive corresponds in our model to the value $E[T_s]$ in the server configuration value.

The scenario is analogous to that one used for the theoretical study. The user traffic has been generated following a Poisson process. A piece of software launches the attack from a single source. Both legal users and intruder traffic flows traverse a WAN network to reach the server, with a round trip time $RTT = \mathcal{N}(17ms, 0.05ms)$. Traces on the users and the intruder side have been issued for collecting the necessary data to calculate the attack indicators.

Among the results obtained from the set of experiments carried out, Table 1 shows some examples with a comparison between the obtained indicators in the simulation and in the real environment, for different mean service times ($E[T_s]$) and user traffic arrival rates ($E[T_a]$). The values for $E[T_a]$ have been selected in such a way that there is no congestion on the server without any attack. In these experiments, we have chosen values for *ontime*= 0.4 s, *interval* = 0.4 s, *offset*= 0, and *Var[T_s]*=0.01. The capacity of the server is $N = 7$.

We can observe the accuracy of the results obtained from simulations when compared with those for the real environment. Surprisingly, we can even obtain better results in efficiency (lower U and higher S, with lower O) for the attack in a real environment in some cases (e.g., first scenario with $E[T_s] = 3$). From these results, two conclusions can be made: a) all the experiments in the simulation case seem to provide results that are good approximations to the behaviour in real environments, and b) the real impact of the attack is very high, showing that these vulnerabilities in monoprocess servers are able to be easily exploited.

6 Conclusions

In this work, a vulnerability present in monoprocess or monothreaded servers is described. It consists in the possibility of knowledge, by an intruder, of the inter-output time of a server under congestion conditions. This vulnerability allows an intruder to perform a denial of service attack against a server. The attack could be designed to nearly or completely saturate the capacity of the target system but, as a difference from the usual brute-force DoS attacks, it uses a relatively low rate of packets to the target server to achieve its goal. Moreover, it is possible to tune the attack parameters in order to select the appropriate values for efficiency and load generated in the server. This distinctive characteristic could allow the attack to bypass, in many cases, existent IDS systems, thus becoming a non-detectable threat.

As a difference to other existent low rate DoS attack [15], this one threatens the application level, maintains the server engaged serving intruder requests and get advantage of the knowledge of the inter-output time of the target server. However, it has some common features with the low rate DoS attack to TCP, what points out the existence of a new family of DoS attacks, characterized by the fact that they rely on vulnerabilities that consist in the a-priori knowledge of one timer of a protocol or end-system behaviour, and that allows the intruder to carry out the DoS attack with a low rate of transmission, avoiding most of the current IDS systems.

The fundamentals and details of the design of a possible exploit have been explained. It has been demonstrated that this attack can be easily carried out and that it can obtain very efficient results. The potential risk that this attack presents is really worrying, due to the fact that it could behave very similar to legacy users, bypassing IDS systems and possibly affecting the services in a server.

As a future work, we plan to extend the study of this vulnerability in concurrent servers, due to the fact that this kind of servers is more widely deployed in Internet.

References

1. M. Williams. Ebay, amazon, buy.com hit by attacks, 02/09/00. IDG News Service, 02/09/00, http://www.nwfusion.com/news/2000/0209attack.html.
2. Jelena Mirkovic , Peter Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Computer Communication Review, v.34 n.2, April 2004
3. CERT Coordination Center, Denial of Service attacks. Available from <http://www.cert.org/tech_tips/denial_of_service>
4. Computer Security Institute and Federal Bureau of Investigation, CSI/FBI Computer crime and security survey 2001, CSI, March 2001. Available from <http://www.gocsi.com>.
5. D. Moore, G. Voelker, S. Savage, Inferring Internet Denial of Service activity, in Proceedings of the USENIX Security Symposium, Washington, DC, USA, 2001, pp. 9-22.

6. P. Ferguson, D. Senie, Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing, in RFC 2827, 2001.
7. Global Incident analysis Center - Special Notice - Egress filtering. Available from <http://www.sans.org/y2k/egress.htm>.
8. X.Geng, A.B.Whinston, Defeating Distributed Denial of Service attacks, IEEE IT Professional 2(4)(2000) 36-42.
9. N.Weiler, Honeypots for Distributed Denial of Service, in Proceedings of the Eleventh IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises 2002, Pitsburgh, PA, USA, June 2002, pp. 109-114.
10. Axelsson S. Intrusion detection systems: a survey and taxonomy. Department of Computer Engineering, Chalmers University, Goteborg, Sweden. Technical Report 99-15; March 2000.
11. R.R.Talpade, G.Kim, S.Khurana, NOMAD: Traffic-based network monitoring framework for anomaly detection, in Proceedings of the Fourth IEEE Symposium on Computers and Communications, 1998.
12. J.B.D. Cabrera, L.Lewis, X.Qin, W.Lee, R.K.Prasanth, B.Ravichandran, R.K.Mehra, Proactive detection of Distributed Denial of Service Attacks using MIB traffic variables - a feasibility study, in Proceedings of the 7th IFIP/IEEE Internation Symposium on Integrated Network Management, Seattle, WA, May 14-18, 2001.
13. J. Mirkovic, G.Prier, P.Reiher, Attacking DDoS at the source, in Proceedings of ICNP 2002, Paris, France, 2002, pp. 312-321.
14. DDoS attacks and defense mechanisms: classification and state-of-the-art, in Computer Networks 44, 2004, pp. 643-646.
15. A. Kuzmanovic and E. Knightly, Low Rate TCP-targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants), in Proc. ACM SIGCOMM 2003, Aug. 2003, pp. 75-86.
16. H.Sun, J.C.S. Lui, and D.K.Y.Yau, Defending Against Low-Rate TCP Attacks: Dynamic Detection and Protection, in Proc. IEEE Conference on Network Protocols (ICNP2004), Oct. 2004, pp. 196-205.
17. G. Yang, M. Gerla, and M. Y. Sanadidi, Randomization: Defense Against Low-rate TCP-targeted Denial-of-Service Attacks, in Proc. IEEE Symposium on Computers and Communications, July 2004, pp. 345-350.
18. A. Shevtekar, K. Anantharam and N. Ansari, Low Rate TCP Denial-of-Service Attack Detection at Edge Routers, in IEEE Communications Letters, vol 9, no. 4, pp. 363-365, April 2005.
19. SANS Institute. NAPTHA: A new type of Denial of Service Attack, December 2000. http://rr.sans.org/threats/naptha2.php
20. Roberta R. Martin. Basic Traffic Analysis. Prentice-Hall Inc. September 1993. ISBN: 0133354075.
21. http://mathworld.wolfram.com/ExponentialDistribution.html
22. Network Simulator 2.Available at: http://www.isi.edu/nsnam/ns/

Delegating Secure Logging in Pervasive Computing Systems

Rafael Accorsi and Adolf Hohl

Department of Telematics,
Albert-Ludwigs-Universität, Freiburg
{accorsi, hohl}@iig.uni-freiburg.de

Abstract. Logging is a central service in computing systems. It lays the foundation for accountability and audit services in computing systems, as well as for other accessory services. While providing logging services in traditional computing systems is a relatively smooth process, it turns to an intricate task in pervasive computing systems. In this context, we present two contributions addressing this problem. First, we develop an approach to securely log information in marginally trusted collectors. Second, we investigate the question of how to securely delegate our logging protocol to a relay equipped with trusted-computing modules.

1 Introduction

Irrespective of the underlying computing paradigm and application, logging services are relevant. Logging services collect and store events communicated by devices within the system, possibly using an intermediary relay, and thereby lay the foundation for accountability and audit services in computing systems. For this to happen, log data must be authentic, for *ex falsum quod libet*, that is, from a falsity everything follows.

Protecting the security of log data, expressed in terms of its authenticity, is thus of foremost importance. In this paper, we focus on pervasive computing systems, i.e., systems allowing for the omnipresent availability of computing power, communication and data.[1] A distinguishing feature of pervasive computing environments is their inherent mixed-mode design, that is, the seamless combination of resource-poor and resource-rich devices: While resource-poor devices lack storage, computing power or energy supply, resource-rich devices do not.

The pervasive setting poses several challenges to security [26], in particular to secure logging. Problems arise when, e.g., the device notifying log events is not the same as the service collecting log data, or when a relay between the device and the collector intermediates the logging process. To counter these problems, two possibilities can be taken into account. One can devise lightweight logging protocols to accommodate the resource limitations imposed by resource-poor devices, or apply to methods to delegate log tasks to external, marginally

[1] We consider this as a result of advances in distinct aspects of computing systems, namely autonomy [1], pervasiveness [23], and reachability [12] of devices.

trusted resource-richer devices offering relay or collector services. The concern behind lightweight methods is the well-known trade-off between performance and security [13], where privileging performance entails in fragile security guarantees. Delegation of log tasks has a similar consequence, as it is non-trivial to devise methods to ensure and obtain evidence that the delegatee carries out the tasks as expected.

In this paper, we report on a logging mechanism to securely store log data in, and delegate log tasks to marginally trusted collectors. Our main contributions are two-fold. Assuming devices' scarceness of storage, we first propose an approach to securely store log data in marginally trusted collectors. The protocol we propose leverages the techniques proposed by [25] and, additionally, provides for non-repudiation of the collector's receipt of log data. Second, assuming that not only storage is scarce, but also the underlying computing power, we investigate the question of how much of our protocol can be (securely) delegated to relay. To this end, we employ trusted-computing platforms to allow for remote attestation and, thereby, to obtain strong security guarantees regarding the behavior of the relay.

Overall, our work sheds a light on the increasingly relevant, yet obscure area of logging and accountability in pervasive computing systems and its combination with secure hardware. At investigating this topic, the first difficulty is the characterization of an appropriate attacker model. Although we sketch such a model, in this paper we merely focus on illegal actions violating the authenticity of log data. To this end, we define what authenticity (and, thereby, security) of log data means and show how it can be harmed.

To distribute log tasks to marginally trusted collectors and relays is not a trivial task. The protocol we propose in §3 assumes that the device has enough computing power to use well-known cryptographic primitives to protect log data and delegate storage to a remote collector. Since this approach is considerably costly and only partially applicable to pervasive computing, we extend our approach with underlying remote attestation techniques based on trusted-computing platforms to delegate the whole process to a relay. As we see below, this considerably saves computation cycles and, as such, could be applied in pervasive computing environments. We remark that, in general, logging mechanisms such as the one we present can also be misused to violate individual's privacy. Although we see privacy as a central issue in pervasive computing environments, we deliberately do not address it in this paper.

This paper is organized as follows: in §2, we introduce the scenario underlying our research into logging and characterize the security properties of log data. We describe our logging protocol in §3, and discuss its adequacy and limitations regarding the expected security properties. In §4 we investigate how to delegate log tasks to marginally trusted relays using trusted-computing platforms, and discuss our results. In §5, we briefly evaluate our approach and suggest alternative approaches to cope with delegation of log tasks. In §6, we report on related work, and draw conclusions and state the upcoming research issues in §7.

2 Preliminaries and Security Properties

We classify the components involved in logging services according to their role. In this setting, a *device* is a service that generates a log message; a *relay* is a service that receives a log message and forwards it to another service; and a *collector* is a service that receives a message and, in one way or another, archives it. We employ the following notation in this paper:

- d_i denotes the i device, r_i the i relay and c_i the i collector; D denotes log data.
- $\{X\}_K$ denotes the encryption of message X under K. $\mathit{Sign}(X)_K$ stands for the signature of X with an asymmetric key K.
- X, X' stands for the concatenation of the messages X and X'.
- K_s stands for the public-key of the service s and K_s^{-1} s's private-key.
- $\mathit{MAC}_K(X)$ denotes the message authentication code of X under K.
- $\mathit{Hash}(X)$ is the one way hash of X.

While we do not prescribe a particular set of algorithms to implement the cryptographic primitives above, we assume that these functions fulfill the expected properties in that, e.g., it is infeasible for an attacker to provoke collisions of hash values

2.1 Characterizing the Security Guarantees

Log data can only be used if it is authentic. We define authenticity as the simultaneous fulfillment of the following properties:

- *integrity*: log data faithfully reflects the reality. That is, the information logged is consistent, accurate, and correct.
- *uniqueness*: log data shall not allow for parallel realities, i.e., it is impossible to intercept log data sent from d_1 to c_1 and to resend it (possibly in modified form and claiming a different device identity) to c_2. This entails confidentiality of log data during the transmission, for log data transmitted in clear can be easily duplicated.

These requirements characterize authenticity properties of log data and are implemented using different cryptographic techniques. Log services based on these techniques need to ensure *tamper evidence*, i.e., attempts to illicitly modify log data must be detectable to a verifier [15], and *forward integrity*, that is, log data contains sufficient information to confirm or rebuke allegations of log data modification before the moment of the compromise [6]. Tamper evidence and forward integrity are necessary since absolute tamper resistance is, in practice, infeasible [5]. This specially holds in pervasive computing systems [28].

2.2 Threat Model and Attacks

The goal of the attacker is to access log data and, thus, violate its integrity and uniqueness. For this, the attacker uses different strategies, which we model using a slightly modified version of the Dolev-Yao attacker model [10].[2] In order to model attacks upon stored log data, we extend the capabilities of the attacker. Namely, once he gets to the logfiles, he can read, delete, and modify stored data. We assume that the attacker cannot misuse cryptography, i.e., he can only perform cryptographic operations using the appropriate cryptographic keys.

Attacks upon log services have different facets and, as such, target different participants. Faithfulness and uniqueness can be attacked using (possibly a mixture of) the following strategies:

– *replay of log messages*: the attacker records a set of messages M sent by a device d to collector c. Later, he attacks d and, in order to hide collected evidence about the attack, sends a (refreshed) set M to c.
– *integrity of sent and stored log data*: the attacker has access to the communication medium and can modify data during the transmission. Moreover, if the attacker gains access to log files at the collector c, he can read and modify them at will.
– *authenticity of device*: the attacker takes over a service, modifies its identity, and starts to send log messages to a collector.
– *confidentiality of log messages*: during the transmission, the attacker intercepts and reads messages sent in clear-text over the network. Similarly, confidential log data stored in a collector may be accessible to an attacker.
– *availability of log service*: the attacker floods the collector c with irrelevant log messages, so that legitimate messages sent by the devices are not logged. The attacker can also use this setting to attack the devices sending messages to c, since his attempts are not protocolized.

3 An Approach to Security Logging

Our method to providing secure logging services in pervasive computing systems builds on and extends the techniques proposed in [25]. To simplify matters, we consider that a device and a collector communicate either without an intermediary relay, or that the relay does not misbehave. Under these assumptions, our protocol consists of the following steps:

1. *mutual authentication of services*: apart from authentication, the device and the collector also agree on a secret value pv_0. This is a proof value used at the acknowledgement phase; its goal is to ascertain authenticity of log messages.
2. *initialization and construction of the logfile*: the device is in charge of applying cryptographic techniques to safeguarding the integrity and uniqueness of its logfile. We assume that chunks of log data are sent from the device to the collector.

[2] We remark, however, that one challenge in pervasive computing systems is the development of an adequate attacker model [28]. For ongoing research on attacker models for pervasive computing systems see [8].

3. *acknowledgement of receipt from collector*: the collector computes a hash value based on the device's messages and sends it signed together with a timestamp and protocol sequence number back to the device. The device then stores this unambiguous piece of information, as it demonstrates that the collector received the chunk of log data correctly and can be held accountable for attacks upon this data.

For our initial aim, we focus on the last two steps; authentication will play a role in §4, where we leave out the aforementioned assumptions and consider an unreliable relay.

3.1 Initializing the Logfile

Assuming that the device and the collector successfully prove their identities to each other and agree on a secret value A_0, d creates the log file by inserting the first entry into it.

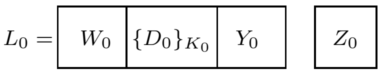


Fig. 1. Initial log entry L_0 and authenticator Z_0

All the entries in the logfile have the same format. We illustrate the initialization entry L_0 in Fig. 1, where the fields stand for the following information:

– W_0 is a permission mask to regulate the access to the log entry L_0. According to [25], at the initialization phase this entry type may be set to **LogfileInitializationType**.
– $\{D_0\}_{K_0}$ is the symmetrically encrypted log data for the entry L_0 and K_0 is a random session key. To provide the necessary security guarantees, D contains not only the event to be logged, but also a timestamp and a protocol identifier. (The former states the actuality of the entry, the latter avoids harmful protocol interactions between different protocol steps.)
– Y_0 stands for the first link of a hash-chain.[3] The actual initialization value of Y_0 is randomly generated.
– $L_0 = W_0, \{D_0\}_{K_0}, Y_0$ is the initial log entry.
– Z_0 is the message authentication code of L_0 defined as $MAC_{pv_0}(L_0)$. This piece of information is used to compute the proof value associated to the whole chunk of log entries and, thence, is *not* send along with L_0 to the collector.

The process of initializing the logfile includes an authenticated response of the collector in order to detect possible tampering.

[3] In the simplest form, a hash-chain Y can be inductively defined as $Y_1 = Hash(Y_0)$ and $Y_n = Hash(Y_{n-1})$.

3.2 Adding Log Entries

After creating the logfile, the device starts adding entries to the logfile. This process encompasses the following operations:

1. $A_j = Hash(A_{j-1})$ denotes the authentication key of the jth log entry. The confidentiality of this information is essential for the security of the log data, as it is used, either directly or indirectly, in every step of the protocol. Thus, we assume that the computation of the new value overwrites irretrievably the previous value.
2. $K_j = Hash(W_j, A_j)$ is the cryptographic key with which the jth log entry is encrypted. This key is based on the permission mask W_j. Thus, only allowed services may access the entry.
3. $Y_j = Hash(Y_{j-1}, \{D_j\}_{K_j}, W_j)$ is the jth value of the hash-chain. Each link of the hash-chain is based on the corresponding encrypted value of the log data. This ensures that the chain can be verified without the knowledge of the actual log entry.
4. $L_j = W_j, \{D_j\}_{K_j}, Y_j$, i.e., the log entry consists of the permission mask, the encrypted log data and the hash-chain value.
5. $Z_j = MAC_{pv_j}(Hash(L_j))$ is the authenticator of the jth log entry, where pv_j is defined as $pv_j = Hash(Z_{j-1}, pv_{j-1})$. Note that we compute the message authentication code for the whole entry instead of a field of it (in the case of Schneier and Kelsey, the hash-chain value Y). While this increases the computational cost involved in calculating of this field, it is necessary to ensure the acknowledgment phase.

This process describes how the jth log entry is computed. We address the overall security guarantees provided by this protocol phase in §3.4.

3.3 Acknowledgement Phase

The last phase of the protocol focuses on the collector and its goal is to provide an irrefutable evidence regarding collector's possession of the chunk of log data sent by the device, as well as chunk's integrity. To this end, the following steps are carried out:

1. by receiving the chunk L_j–L_k (with $j < k$), the collector computes for each entry L_i the corresponding A_i and Z_i values.
2. after $k - j$ iterations, the collector gets obtains $MAC_{pv_k}(Hash(L_k))$.
3. the signed proof value $Sign(MAC_{pv_k}(Hash(L_k)))_{K_c^{-1}}$ is sent to the device. This message includes a timestamp and protocol step identifier.
4. the device then checks whether the authenticator matches with the authenticator computed during the second phase. If yes, the devices frees the storage by deleting the chunk.

This concludes the protocol. The device can now ascertain the integrity of the logfile, as well as (selectively) delegate access to its entries.

Table 1. Devices' computational cost associated to running the logging protocol

Protocol phase	**Kind of cryptographic operation**				
	PSG	MAC	Hash	SC	AC
Initialization	3	1	–	1	–
Appending an entry	–	1	4	1	–
Acknowledgement	–	1	–	–	1

On the Computational Cost. In pervasive computing systems, the applicability of an approach depends on the computational effort involved in running its implementation. To this end, we use the number of cryptographic operations carried out by the device while running the protocol, as shown in Table 1, where PSG stands for pseudo-random generation, SC for symmetric cryptography, and AC for asymmetric cryptography. For certain resource-poor devices, this effort can be an overkill. In §4, we present an approach to delegate these task to a relay, and compare the associated effort.

3.4 Discussion on the Required Security Guarantees

Assuming that the device accurately archives the events its sensors communicate, guarantees regarding integrity are safeguarded by the proof value sent by the collector in the last protocol phase. The message authentication code employed by the collector proves that it possesses the corresponding secret A and demonstrates that every log entry is a faithful representation of the original log data. It should also be noted that although the collector possesses the secret A, it cannot read the log data D sent by the device, as it is sent in encrypted form.

This fact is closely related with the concept of forward integrity. If an attacker takes over a collector c at time t, he is able to act as if c were not compromised. For this, he merely keeps sending the proof values as prescribed by the protocol. But, since he cannot obtain the previous values of A, it is impossible for him to read log entries archived before t. Thus, these entries fulfill confidentiality and integrity properties.

Regarding uniqueness, assuming that the cryptographic functions employed in the algorithms are sound, uniqueness is guaranteed by the use of timestamps and the hash-chains. Timestamps bring along a synchronization problem between the device and the collector. Although the probability of an attack due to timing confusion might be negligible in practice, the hash-chain works in a complementary manner and prevents old messages from being seamlessly and undetectably added to the chain.

This leads to the need for tamper evidence. The hash-chain employed in the protocol lets us verify the integrity of the whole chain by inspecting the last link in the chain. The proof value send by the collector demonstrates that the log data has not been tampered with during the transmission or by a malicious process acting between the device and the collector. Hash chains also make it possible to detect whether (series of) log entries were eliminated.

3.5 Limits of the Logging Protocol

The logging protocol we present above is based on a number of assumptions, which outline and complement the underlying technical restrictions. We now briefly discuss the most significant limitations.

- Timepoint of a successful attack: the integrity of the logfile can only be ensured up to the point in time at which the attacker takes over the collector (or device). Once an attacker succeeds in compromising a machine, he can misuse log information at will.
- Deletion of log entries: it is always possible for an attacker to completely (and irretrievably) delete the log file or entries thereof, in particular when we assume that no write-only hardware or media (e.g. WORM disk, paper print-out, optical jukebox) is at hand. Tamper evidence allows us to detect the deletion.
- Collector's behavior: although the device receives an evidence that the collector appropriately receives the log data, it cannot be sure that it stores the data. To obtain this kind of security guarantees, below we employ trusted computing-based techniques (see §4).

4 On Delegation of Tasks

In pervasive computing environments, devices with different resource constraints participate. Resource constraints such as a slow computing power caused by slow CPU or a battery-based power supply are reasons for delegating the processing of resource intensive tasks to resource-rich devices. In this case, a service on the relay can provide suitable services with the required protection goals of the log data. Such a shift of computation tasks to a virtual coprocessor is visualized in Fig. 2. A resource-poor device buys computing power and services from a powerful platform for outsourcing the computing intensive tasks.

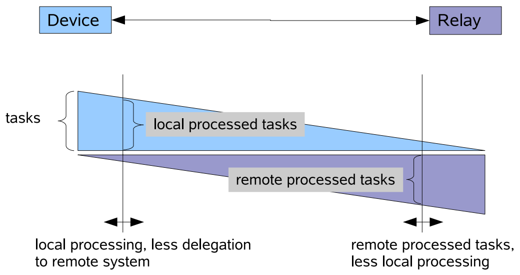

Device
Relay
tasks
local processed tasks
remote processed tasks
local processing, less delegation to remote system
remote processed tasks, less local processing

Fig. 2. Shift of computation tasks from a delegator to a delegatee

To central requirement for a successful delegation is the absence of negative side-effects regarding the underlying security guarantees. On the other hand, the necessary requirements for a virtual coprocessor are:

- *secure communications*: it must be possible to establish a secure connection to the virtual and remote coprocessor for confidential data communication. From the perspective of a s resource-poor device, a resource effective security protocol is necessary.
- *exclusiveness*: the exclusiveness of the virtual coprocessor is necessary to prevent any side effects of other programs and actions on a system with the virtual coprocessor.
- *authentication*: to establish a trust relation between the logging device and the virtual coprocessor, the device authenticates selected attributes at the virtual coprocessor. Although device based authentication works fine when the device and the service are in the same administrative domain, problems arise when they are in distinct ones, as it could be in pervasive environments. Therefore, similar to Creese et al. [9] attributes of the service should be authenticated instead. Because of the *Undecidability Theorem* it is not possible to derive the behavior of a piece of code in general, the authentication of behavior could be done via code identification and a suitable mapping. In this case, it would be no longer necessary to authenticate its identity and rely on trust relationships of their maintainers.

4.1 Building an Adequate Coprocessor

A Trusted Computer (TC) platform provides an important building block for a virtual coprocessor. Trusted Computing technology [27] is widely available and is used below to outsource the computations for secure logging from a resource-poor device to a in general marginally trusted relay. TC platforms provide functionality to faithfully transmit attributes of a platform to a remote challenger via the so-called *remote attestation*. This functionality of TCG-compliant platforms allow the authentication of executed code via a Trusted Platform Module (TPM). The TPM acts as a secure logbook of executed code stores their hashes in the so-called platform configuration registers (PCR) for an audit of the platform later.[4] Therefore, it is the duty to report every code hash from the startup of the platform. The values in the PCRs can be communicated faithfully and by mapping the authenticated code fragments to a set of properties of the system, one can decide, whether the system is secure in his sense.

With the use of a TC-Platform, a virtual coprocessor could be easily authenticated. For the delegation of a logging task, the relay system has to prove the existence of an operating system [20] with a resistant process isolation to provide exclusiveness for a virtual coprocessor. Therefore, the relay has to run a system with this attribute which is known by the device. Furthermore, for a secure communication, avoiding any time delay attacks when authenticating a relay using remote attestation is used [14].

[4] The TCG specifications up to 1.1b prescribe 16 registers with 160 bit; TCG 1.2 specifies at least 24 registers.

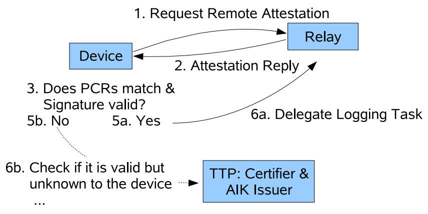


Fig. 3. Remote attestation of the relay

4.2 Delegation of Secure Logging Task

If the behavior of a service could be authenticated, a service doing the job as expected can be selected and authenticated. Although Trusted Computing does not allow to authenticate a certain behavior, it supports the authentication of a binary which behaves as expected. Taking for granted that the service is not interfered by other activities on the relay, this one could be seen as a virtual secure coprocessor of the device. In this case, the secure logging task can be delegated to the remote relay lying not necessarily in the same administrative domain.

In the scenario, we assume a resource poor device. It would be advantageous to use a simple representation of valid relay configurations to enable a fast authentication of the virtual coprocessor (including the platform which provides it). Therefore, we propose to represent the platform configuration by a few sets of valid relay configurations. We do not pay attention to the history which leaded to this configuration. This has the consequence, that other valid configurations are excluded.[5] Assuming a static platform configuration, the same functionality can be achieved using the so-called *sealing* functionality.

To establish a the secure communication, meaning a device can authenticate a qualified relay when transmitting log data an authentication of the corresponding service access point of the log daemon has to be performed. The relay transmits the hash of a generated authentication secret. Later on, this secret is used to authenticate a prior platform a remote attestation was requested from. One selected PCR is used to carry the fingerprint of this authentication secret to get it signed during the remote attestation procedure. Thus, we misuse a PCR as a signing channel using a so-called *attestation identity key* (AIK).[6] This saves a further key pair and certificate for signing purposes.

The device then authenticates a virtual secure coprocessor via remote attestation. A signed set of the PCR is replied. If they match with a known set of

[5] We remark that different calling orders of code usually result in other values in the PCR of the TPM.

[6] One of many certificates to sign PCRs, also known as platform identity. They are issued based on whether the platform is a valid TCG system.

values (representing an untampered virtual coprocessor), the device proceeds. Otherwise the device needs to contact a trusted third-party to acquire information about acceptable relay configurations [21].

The delegation of the logging task requires the following steps and computational costs for the device:

1. *authentication of the virtual coprocessor*: the relay authenticates itself with its platform configuration. Hence, every relay with the same platform configuration is also qualified for a the delegation of the logging task. The platform configuration consists of the PCRs. These registers are signed during remote attestation. The computational cost for the device is to compute the hash of the transmitted registers and perform the verification of a RSA signature.
2. *authentication of its service access point*: the service access point of the virtual coprocessor has to be authenticated. For this step, a selected PCR is used to transmit the hash of a public key (previously generated) of the secure coprocessor during the remote attestation procedure before. When the device connects to the relay, the relay presents the public key. With that, the device can verify the signed hash and the former platform configuration belongs to this relay. Then the device performs a challenge which only could be answered correctly by the corresponding relay. While [14] uses a RSA key pair, an algorithm suitable for resource-poor devices should be selected. The computational cost for the device is to perform a hash of the public key and encrypt a challenge in either RSA or a more computational efficient asymmetric cipher.
3. *setup of a secure channel*: the challenge sent in the step before can be used as a key for a symmetric cipher. Both, the device and the relay know that key solely and can communicate in a confidential way using any protocol which provides a symmetric encryption.

For each log entry the device transmits data to the relay. Additional computation arise for the encryption. Assuming that a log entry has an overall length of about 100 bytes, about 13 cycles with a symmetric block cipher (with 8 byte per block) are necessary to encrypt the log data. Table 2 summarizes the necessary cryptographic operations of the device when the logging task is delegated.

Table 2. Devices'computational cost associated to delegate the log task

Protocol phase	**Kind of cryptographic operation**				
	PSG	MAC	Hash	SC	AC
Initialization	1	–	1	–	2
Transmitting an entry	–	–	–	1	–

5 Discussion and Evaluation

We now briefly compare the effort involved in using the protocol we propose, both with and without delegating the cryptographic operations involved in the protocol.

The initialization phase of the protocol consumes considerably less resources in the case without a TC platform, as it does not include asymmetric RSA encryption. Asymmetric RSA is necessary, since this algorithm is used for signing the attestation vector. Transmitting an entry is, according to our analysis, cheaper than appending it, as it only requires a symmetric encryption of log entries. Also, using the TC platform it is not necessary to add an acknowledgement phase; this could be handled with the delegatee, i.e., the relay.

Although computing power rises fast, it would be adequate for pervasive computing devices with low power and computing resources to save computing cycles and use other algorithms than RSA. Such algorithms are already available, e.g., multi exponent RSA, XTR or ECC [7]. While this specific issue could not be addressed with TCG platforms, the promising *XOM* (execute only memory) could be used [17]. In XOM, the attestation of an application is handled inside the application. The software designer has the freedom to select an algorithm which suits best. This would cut down the necessary computing cycles a device needs for an authentication procedure of the delegated logging service. Unfortunately, the XOM approach is in an academic stage and not available, therefore the TCG approach is used instead.

6 Related Work

A number of approaches have been proposed for securely logging information in computing systems. The majority of these approaches are based on syslog, a de facto standard log service whose functionalities are described in the RFC 3164 [18]. Security was not considered at developing syslog: it provides no device authentication; it ignores the original source of log data; it transmits and stores log data in clear-text; and uses UDP as underlying transport protocol. Security was thus an afterthought. Indeed, secure extensions have been developed.

– syslog-ng: syslog's new generation aims at providing a secure, reliable log service [4]. It is backward compatible with RFC 3164 and its only advantage against syslog is the use of TCP as transport protocol. Encrypted and signed transmission and storage are still not provided.
– syslog-sign: This service adds origin authentication, message integrity, replay resistance, and detection of missing messages to syslog [16]. This is accomplished with a cryptographically signed message that contains the signatures of previously sent syslog messages. The contents of this special message is called "signature block." To our knowledge, no implementation of syslog-sign is available to-date. syslog-sign does not provide confidentiality during the transmission of log data, nor does it account for encrypted storage of log

entries. Moreover, since the signature blocks may be deleted after the authentication, tamper evidence and forward integrity are only partially fulfilled.
- syslog-pseudo: Logfiles often store personal data and, thus, are a popular attack point. syslog-pseudo proposes an architecture to sanitize logfiles in unix-like systems [11]. This approach focuses exclusively on the privacy of users and does not take into account the security properties suggested in §2.1.
- Reliable syslog: While previous approaches are based on the RFC 3164, the reliable syslog is based on the RFC 3195 [19] and aims to implement reliable delivery for syslog messages. For this, it is built on top of BEEP, the Block Extensible Exchange Protocol [2]. BEEP is a framework for building application protocols that uses TCP and allows device authentication, mechanisms to protect the integrity of log messages, and protection against replay attacks. An implementation of reliable syslog is available [3].
- Schneier and Kelsey: This method is the starting point of our investigation. In essence, it aims at realizing tamper evidence and forward integrity. In addition to that, mechanisms to protect the integrity of log entries, as well as their confidentiality and access control are presented [25]. However, the assumptions underlying our approach, as well as the procedures involved in achieving the security guarantees, differ in various ways. The authors assume that the device and the collector are the same machine. They also assume that the owner of a device is not the same person as the owner of the secrets within the device. In this context, log services must be in place to determine whether there has been some fraud. To our knowledge, this method has not been implemented.

The use of trusted coprocessors to support secure logging has also been explored by Schneier and Kelsey [24]. In contrast to our work, they use trusted coprocessor to authenticate the source of log data but do not support delegation of the logging task.

Concerning trusted computing, current research focuses on minimal trusted computing bases and the integration of TCG compliant platforms. Promising work is done by Pfitzmann et al. [20]. They seperate program execution by compartments, implemented on a micro kernel architecture. The interpretation of remote attestations is the concern of Sadeghi et al. [22]. They map values of PCRs and execution history of a TC platform to properties of a platform. In our scenario, applying this approach could lead to more platform configurations of a relay, accepted by a device.

7 Conclusion and Outlook

We have proposed an approach to securely distribute log services among marginally trusted collectors and relays. Our approach combines standard cryptographic techniques and secure hardware-based technologies to lay a fundamental basis for accountability and audit in pervasive systems. Our work sheds a light on, and allows us to characterize the difficulties of, deploying log services in pervasive computing environments.

This is merely the first step and many interesting research questions are still open. First, the permission mask can be used by authorized services—e.g. audit or configuration services—to obtain log data directly from a collector. For this, protocols to regulate delegation should be in place and analyzed against the protection goals presented in §2.1 in order to avoid harmful protocol interactions. We plan to examine these issues and develop protocols for these purposes.

Second, an adequate attacker model for pervasive systems is still lacking. Such a model should also take into account the mixed-mode setting of pervasive computing systems and its underlying features. At our institute, research is being carried out in this direction and we plan to investigate different attacker models and attacks against our protocol.

Third, although the initial experiments using our logging protocol are promising, we do not have a practical handle on the complexity involved in running the protocol. An in-depth analysis should reveal the extent to which it can be applied in pervasive systems and, by this means, show when a delegation-based approach is necessary. This result is fundamental to the applicability of our approach.

Finally, if misused, such a powerful log mechanism is a privacy *endangering* technology, specially in an environment where virtually *every* piece of information can be collected and processed in order to derive further characteristics of an individual. We plan to address the non-technical implications caused by our approach to logging.

References

1. Autonomic computing. `http://www.research.ibm.com/autonomic/`, 2005.
2. BEEP. `http://www.beepcore.org`, 2005.
3. Reliable syslog. `http://security.sdsc.edu/software/sdsc-syslog/`, 2005.
4. Syslog-ng web site. `http://www.balabit.com/products/syslog_ng`, 2005.
5. R. Anderson and M. Kuhn. Tamper resistance: A cautionary note. In *2nd USENIX Workshop on Electronic Commerce*, pages 1–11. USENIX Assoc., 1996.
6. M. Bellare and B. Yee. Forward integrity for secure audit logs. Technical report, Univ. of California at San Diego, Dept. of Computer Science & Engineering, 1997.
7. E.-O. Blaß and M. Zitterbart. Towards acceptable public-key encryption in sensor networks. In *IWUC*, pages 88–93. INSTICC Press, 2005.
8. S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting empirical engagement in authentication protocol design. In volume 3450 of *LNCS*, pages 119–133. Springer, 2005.
9. S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for pervasive computing. In volume 2802 of *LNCS*, pages 116–129. Springer, 2003.
10. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.
11. U. Flegel. Pseudonymizing unix log files. In volume 2437 of *LNCS*, pages 162–179. Springer, 2002.
12. G. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, 1994.
13. M. Graff and K. van Wyk. *Secure Coding: Principles & Practices.* O'Reilly, 2003.
14. A. Hohl, L. Lowis, and A. Zugenmaier. Look who's talking: Authenticating service access points. In volume 3450 of *LNCS*, pages 151–162. Springer, 2005.

15. G. Itkis. Cryptographic tamper evidence. In *Proceedings of the Conference on Computer and Communication Security*, pages 355–364. ACM Press, 2003.
16. J. Kelsey and J. Callas. Signed syslog messages. IETF Internet Draft, 2005. `http://www.ietf.org/internet-drafts/draft-ietf-syslog-sign-16.txt`.
17. D. Lie, C. A. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. C. Mitchell, and M. Horowitz. Architectural support for copy and tamper resistant software. In *ASPLOS*, pages 168–177, 2000.
18. C. Lonvick. RFC 3164: The BSD syslog protocol. 2001. `http://www.ietf.org/rfc/rfc3164.txt`.
19. D. New and M. Rose. RFC 3195: Reliable delivery for syslog. 2001. `http://www.ietf.org/rfc/rfc3195.txt`.
20. B. Pfitzmann, J. Riordan, C. Stüble, M. Waidner, and A. Weber. Die PERSEUS Systemarchitektur, 2001.
21. J. Poritz, M. Schunter, E. V. Herreweghen, and M. Waidner. Property attestation: Scalable and privacy-friendly security assessment of peer computers. Technical Report RZ3548, IBM Corporation, 2004.
22. A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *Proc. of the 2004 Workshop on New Security Paradigms*, pages 67–77, 2005. ACM Press.
23. M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, August 2001.
24. B. Schneier and J. Kelsey. Remote auditing of software outputs using a trusted coprocessor. *Future Generation Computer Systems*, 13(1):9–18, July 1997.
25. B. Schneier and J. Kelsey. Security audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, May 1999.
26. F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, 2002.
27. Trusted Computing Group. TCG Backgrounder, May 2003.
28. J. Wang, Y. Yang, and W. Yurcik. Secure smart environments: Security requirements, challenges and experiences in pervasive computing. In *NSF Pervasive Computing Infrastructure Experience Workshop*, 2005.

Implementing Minimized Multivariate PKC on Low-Resource Embedded Systems

Bo-Yin Yang[1,⋆], Chen-Mou Cheng[2], Bor-Rong Chen[2], and Jiun-Ming Chen[3]

[1] Dept. of Math., Tamkang University, Tamsui, Taiwan
by@moscito.org
[2] Harvard University
{doug, brchen}@eecs.harvard.edu
[3] Chinese Data Security, Inc., and Nat'l Taiwan U., Taipei
jmchen@math.ntu.edu.tw

Abstract. Multivariate (or $\mathcal{MQ}$) public-key cryptosystems (PKC) are alternatives to traditional PKCs based on large algebraic structures (e.g., RSA and ECC); they usually execute much faster than traditional PKCs on the same hardware. However, one major challenge in implementing multivariates in embedded systems is that the key size can be prohibitively large for applications with stringent resource constraints such as low-cost smart cards, sensor networks (e.g., Berkeley motes), and radio-frequency identification (RFID). In this paper, we investigate strategies for shortening the key of a multivariate PKC. We apply these strategies to the Tame Transformation Signatures (TTS) as an example and quantify the improvement in key size and running speed, both theoretically and via implementation. We also investigate ways to save die space and energy consumption in hardware, reporting on our ASIC implementation of TTS on a TSMC 0.25μm process. Even without any key shortening, the current consumption of TTS is only 21 μA for computing a signature, using 22,000 gate equivalents and 16,000 100-kHz cycles (160 ms). With circulant-matrix key shortening, the numbers go down to 17,000 gates and 4,400 cycles (44 ms). We therefore conclude: besides representing a future-proofing investment against the emerging quantum computers, multivariates can be immediately useful in niches.

Keywords: Multivariate public-key cryptosystem, efficient implementation, digital signature schemes, embedded system, sensor networks, motes.

1 Introduction

A multivariate public-key cryptosystem (a multivariate PKC, an $\mathcal{MQ}$-scheme, or simply a multivariate) uses as the public key the $mn(n+3)/2$ coefficients of a quadratic polynomial map $V = (\ell_1, \dots, \ell_m) : K^n \to K^m$ with no constant term, where $K = \mathrm{GF}(q)$ is a finite field, called the "base field". Computing $\mathbf{w} = V^{-1}(\mathbf{z}) = (w_1, \dots, w_n) \in K^n$ is considered a hard problem, generically NP-hard [16].

[⋆] Partially sponsored by Taiwan's Nat'l Science Council grant NSC-94-2115-M-032-010 and Taiwan Information Security Center (TWISC) at Nat'l Taiwan U. of Sci&Tech, Taipei, Taiwan.

In practice, the trapdoor needed for a PKC is almost always accomplished by using a public map composed of three maps as in $V : \mathbf{w} \in K^n \overset{\phi_1}{\mapsto} \mathbf{x} \overset{\phi_2}{\mapsto} \mathbf{y} \overset{\phi_3}{\mapsto} \mathbf{z} \in K^m$. The *central map* ϕ_2 is quadratic. $\phi_1 : \mathbf{w} \mapsto \mathsf{M}_1\mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathsf{M}_3\mathbf{y} + \mathbf{c}_3$ are affine, usually invertible. The private key is the $m(m+1) + n(n+1)$ coefficients in $(\mathsf{M}_1, \mathsf{M}_3, \mathbf{c}_1, \mathbf{c}_3)$ plus whatever necessary to compute ϕ_2^{-1}.

The security of a multivariate PKC thus depends on the infeasibility of decomposing maps and that of solving a large system of polynomial equations. The reader is referred to [41] for a comprehensive reference for multivariate PKCs, including the nomenclature and the state of the art.

1.1 Advantages That Accrue to Multivariates

When quantum computers (QCs) with thousands of qubits arrive, RSA and discrete-log schemes will be broken [40]. $\mathcal{MQ}$- and lattice-type schemes stand with halved log-complexity ([22], e.g., $2^{200} \to 2^{100}$). We also have quantum key exchange but not quantum signatures. So $\mathcal{MQ}$-schemes seem like future-proofing insurance.

Another reason is that the private map of RSA is intrinsically slow. As a result, transaction speed on commodity hardware suffers, hindering wider deployment. This slowness is magnified on embedded systems [1, 44], where lack of megahertz hurts multivariates less because these schemes spend most of their time in lookup tables.

1.2 Challenges for Multivariates and Our Contributions

Currently, multivariate schemes face some major challenges:

Novelty: This leads to distrust and a prolonged lack of interest. Consequently, there is little in the way of provable security results or optimizations as for RSA or ECC.

Key sizes: Multivariates have a large public key and sometimes a large private key as well. This necessitates costlier hardware just like the slow private map of RSA.

We will herein discuss aspects of low-resource implementations. We will illustrate with the signature scheme TTS (Sec. 2), *but the following apply to most $\mathcal{MQ}$-schemes if one wishes to cut down the private key:*

Key scheduling lets a smart card carry a small "mini key" (Sec. 3), based on which the "real key" and the private map is built on the fly.

Structure in the linear maps enables representation of M_1 and M_3 with fewer parameters. We can have a private key that is around 1/5 of its original length and use 40% less running time (Sec. 4).

Despite the fact that most multivariate schemes are already fairly fast, and that the public keys are much longer (which makes them the natural place to start if one wishes to cut down memory usage), we feel that our effort is justified in view of the following:

- Often a scheme needs to be as fast as possible, not just "fast enough".
- In many applications, we only need to store the private key on-card; for example, when the embedded devices are used as a cryptographic token.

– Some applications can tolerate lower security but absolutely must fit within very tight resource limits. RSA simply won't fit on a low-end RFID. But a multivariate PKC might if its private key can be shortened. Schemes operating on small chunks are more likely to be implemented in such an environment [15].
– The central map of an $\mathcal{MQ}$-scheme often determines its properties, including weaknesses. *A chain can only be as strong as its weakest link.* So if a given central map leads to a cryptanalysis in, say, 2^{100}, then the 100+ parameters in the matrices are more likely to be an overkill rather than a security enhancement.

The rest of this paper is organized as follows. In Sec. 2, we go over the Tame Transformation Signatures (TTS), a workbench for key-shortening experimentation. Subsequently in Sec. 3, we discuss the aspects of key scheduling, and in Sec. 4, we describe the key-shortening strategy via restructuring the linear maps. In Sec. 5, we report the result of our application-specific integrated circuit (ASIC) implementation, mainly targeting at RFID applications. We conclude with the discussion in Sec. 6.

Some of the diversed results referenced are summarized in the appendix; the rest needs to be looked up from the original sources [8, 10, 11, 13, 27, 37, 41, 43, 44].

2 Tame Transformation Signatures

Before we move on, we need to emphasize that there is no reductionist proof of security today; the security of $\mathcal{MQ}$-schemes is so far thrust-and-parry. Since this is mainly about implementation, we leave out most details about security, although we check that known attacks are not functional against our schemes. The interested reader is referred to Appendix A, where we report the result of a few basic cryptographical experiments.

2.1 Classification of Multivariates, and enTTS (20,28)

In designing and implementing key-shortening techniques for multivariates, we note that there is a rationale to experimenting with variants of existing ones instead of inventing new ones. Extant schemes represent accumulated experience and history. In tweaking them appropriately, one can expect to inherit many of the good properties and reuse code, which introduces fewer opportunities for mistakes.

There are four basic ways [41] one can set up the trapdoor of a multivariate scheme for ϕ_2 to be inverted: Imai-Matsumoto's C^* [31] (and derivatives SFLASH [39] and PMI+ [10]), Hidden Field Equations [37], "Unbalanced Oil-and-Vinegar" [27], and "Stepwise Triangular System" [43]. Basic trapdoors must be modified for security — see [41] for a summary of modifications. All have $O(n^3)$-time (and space) public maps in the practical range, where n is the size of the digest or plaintext block.

C^* and HFE are "dual-field" multivariates, which operate over a larger field $K^k \equiv \mathrm{GF}(q^k)$. UOV and STS are "single-field" multivariates. The TTS [43] family of signature schemes use a UOV-STS combination. We will illustrate most of our techniques on a current variant in the family called the Enhanced TTS with $K = \mathrm{GF}(2^8)$, 20-byte hashes, and 28-byte signatures, also known as enTTS (20,28). This has the central map

$\phi_2 : \mathbf{x} = (x_0, x_1, \ldots, x_{27}) \mapsto \mathbf{y} = (y_8, y_9, \ldots, y_{27})$:

$$\begin{aligned}
y_i &= x_i + \textstyle\sum_{j=1}^{7} p_{ij} x_j x_{8+(i+j \bmod 9)}, \; i = 8 \cdots 16;\\
y_{17} &= x_{17} + p_{17,1}x_1x_6 + p_{17,2}x_2x_5 + p_{17,3}x_3x_4\\
&\qquad + p_{17,4}x_9x_{16} + p_{17,5}x_{10}x_{15} + p_{17,6}x_{11}x_{14} + p_{17,7}x_{12}x_{13};\\
y_{18} &= x_{18} + p_{18,1}x_2x_7 + p_{18,2}x_3x_6 + p_{18,3}x_4x_5\\
&\qquad + p_{18,4}x_{10}x_{17} + p_{18,5}x_{11}x_{16} + p_{18,6}x_{12}x_{15} + p_{18,7}x_{13}x_{14};\\
y_i &= x_i + p_{i,0}x_{i-11}x_{i-9} + \textstyle\sum_{j=19}^{i-1} p_{i,j-18}\, x_{2(i-j)-(i \bmod 2)}\, x_j + p_{i,i-18}x_0x_i\\
&\qquad + \textstyle\sum_{j=i+1}^{27} p_{i,j-18}\, x_{i-j+19}\, x_j, \; i = 19 \cdots 27.
\end{aligned}$$

Given message M, we compute a 160-bit secure hash digest vector $\mathbf{z} = H(M)$ from the message. We may now compute $\mathbf{y} = \mathsf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$, then $\mathbf{x} \in \phi_2^{-1}(\mathbf{y})$ as follows:

1. Assign random $x_1, \ldots, x_7$ and try to solve the first 9 equations for x_8 to x_{16}.
2. Solve serially for x_{17} and x_{18} using the next two equations (y_{17} and y_{18}).
3. Assign a random x_0 and try to solve the last 9 equations for x_{19} through x_{27}.

We find the signature $\mathbf{w} = \mathsf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$. Release $(M, \mathbf{w})$. At most 9 values of x_7 and x_0 make the two systems unsolvable, so we will find a solution.

enTTS (20,28) has 8680- and 1417-byte public and private keys. A smaller instance enTTS (16,22) has this central map, along with 4400- and 879-byte public and private keys [43]:

$$\begin{aligned}
y_i &= x_i + \textstyle\sum_{j=1}^{6} p_{ij} x_j x_{6+(i+j+1 \bmod 7)}, \; i = 6 \cdots 12;\\
y_{13} &= x_{13} + p_{13,1}x_1x_4 + p_{13,2}x_2x_3\\
&\qquad + p_{13,3}x_7x_{12} + p_{13,4}x_8x_{11} + p_{13,5}x_9x_{10};\\
y_{14} &= x_{14} + p_{14,1}x_2x_5 + p_{14,2}x_3x_4\\
&\qquad + p_{14,3}x_8x_{13} + p_{14,4}x_9x_{12} + p_{14,5}x_{10}x_{11};\\
y_i &= x_i + p_{i,0}x_{i-7}x_{i-9} + \textstyle\sum_{j=15}^{i-1} p_{i,j-14}\, x_{2(i-j)-(i \bmod 2)}\, x_j + p_{i,i-14}x_0x_i\\
&\qquad + \textstyle\sum_{j=i+1}^{21} p_{i,j-14}\, x_{i-j+15}\, x_j, \; i = 15 \cdots 21.
\end{aligned}$$

This was built to compare to RSA-768, while enTTS (20,28) was to RSA-1024.

2.2 History and Security of enTTS and Other TTS Schemes

The STS-UOV family of $\mathcal{MQ}$-signatures contains in addition the Rainbow and TRMS schemes [2, 11]. The combination of recursive segmented evaluation and solving consecutive linear systems defend against the three well-known attacks based on linear algebra against STS [21] and UOV [27]. TRMS uses intermediate-field arithmetic, which also in effect solves a linear system. Both TRMS and enTTS are *sparse* variants, i.e., a TTS-type scheme, with fast signatures and key generation on both PCs and smart cards. The schemes in [44] slightly differ from enTTS due to the UOV-based attacks carried out by Ding and Yin [12], while high speed on a smart card is still retained.

The enTTS (20,28) scheme was designed to account for all known attacks [43], including linear algebra attacks mentioned above, algebraic attacks based on Faugère's

$\mathbb{F}_5$ and XL of Courtois *et al* [8, 13], improved search methods [4], and some methods tailored to specific schemes [19, 20, 36]. We repeated experiments checking that no vulnerabilities had been reintroduced in our short-key versions.

2.3 Performance on State-of-the-Art Sensor Nodes

To evaluate the performance of enTTS on state-of-the-art sensor nodes, we benchmark enTTS on one of the most popular wireless sensor network devices—the Tmote Sky mote. Tmote Sky is equipped with an 8 MHz Texas Instrument MSP430 microcontroller and a Chipcon CC2420 2.4 GHz radio that supports IEEE 802.15.4 wireless low-power medium access control standard. Running on top of it is the TinyOS sensor network operating system [23]. TinyOS is a modular event-driven operating system designed specifically for sensor network applications. It has a component-based programming model, supported by the nesC language [18]. Applications are written as modules that are composed together through a set of predefined interfaces, which describe the events to be handled and the implemented commands. The composition of modules is achieved in compile time by the nesC compiler, whose output is a single C source program, subsequently to be compiled by a regular C compiler into the MSP430 binary code. It is then downloaded onto the sensor devices to run. Such TinyOS-based wireless sensor systems are under active study and development by the sensor network community; they are also becoming a candidate for building pervasive computing infrastructures.

Security and privacy issues, such as authentication and data integrity, are important in some of these emerging applications, e.g., patients vital signals monitoring and dissemination in a hospital. Unfortunately, the more traditional PKCs have not been able to apply in these applications, due largely to the limited computational resources available on the sensor nodes. This is where $\mathcal{MQ}$ schemes like enTTS can find immediate applications.

A typical sensor node like the Tmote Sky mote has a small working RAM (10 KBytes in this case), a slightly larger read-only program memory (48 KBytes), and a relatively large flash memory (1 MBytes) for storing collected raw data and other auxiliary information for its operations. In our case, the private keys are small and can be fit into the working RAM. This helps achieving high signature performance, which is important because it is usually these computationally challenged sensors that are signing the raw data to protect data integrity. For verification, we need to store the public key in flash memory because they are too big to fit into RAM. We argue that this is tolerable because verification is mostly done in the relatively more powerful base stations, although it is needed occasionally by the motes, say, when they need to verify the authenticity of a message coming from a base station or another mote.

We benchmark both the verification and the signature performance of enTTS (20,28): The average signing time is 71 ms, while the average verification time is 726 ms[4], measured with the time provided by TinyOS at the granularity of 1/32,768 seconds, averaging over 1,000 runs. We note that the throughput of flash memory reads on our Tmote Sky mote is merely around 45 KBytes per second, so we spend a

[4] These numbers do not include the time in computing message digests.

significant portion of the verification time reading the 8,680-byte public key off the flash memory. We believe that the verification time can be significantly improved if the entire public key can be stored in a larger working RAM, which is expected to be in the future releases of the device. Even with such an impediment, the results are quite promising compared with previous results obtained using more traditional PKCs such as ECC [30].

3 From Previous Attempts at Key Shortening to Scheduling

Many attempts to use subfields for smaller keys in multivariates have been made. The original version of SFLASH used a subfield but was broken [19]. In general, using subfields to cut down public key size is not a very good idea. The reason is that, in many of the attacks against multivariates such as those in [21, 27, 38], guessing at variables is needed; thus the security is highly correlated to the size of the field in which the matrices elements of M_1 and M_3 are taken.

However, using elements of subfields for parameters in the central map *can* cut down private key size as will be in use in an implementation below. Further, subfield constructions for hardware (especially ASIC) are common. Especially notable are the layered designs of C. Paar [34, 35], which we borrowed into our designs here.

3.1 Key Scheduling and Its Practical Considerations

In general, key scheduling (generating sub-keys or round keys from an original key) will not shorten the running time. In fact, it usually makes everything slower. But it is considered worthwhile because short keys have advantages. In presenting SFLASH^{v2} [39], the authors mentioned that it is possible to run SFLASH with a key schedule. We try to affirm this idea with an actual run and to provide a control.

We have a few obvious candidates. AES submissions have well-tested key schedules with assembly-language routines available to call on the PC and the 8051, so we can use the Rijndael key set-up routine. Or we can also use a stream cipher like RC4 (with a variable key size), or a fast linear feed-back pseudo random number generator (PRNG) such as TT800 (a precursor to the mersenne twister, up to 100-byte keys, cf. [32]).

It seems natural to treat each routine as a PRNG or stream cipher. Two observations:

- We store sub-keys (random numbers) in a buffer and take inputs as needed.
- If we wish to output the public key on demand, we must be able to evaluate ϕ_i and ϕ_i^{-1} in any order. This is very difficult unless the state involved is very short, since we may have to keep six or seven sets of state for each stage.

A trick from [44] seems applicable: Factor both M_1 and M_3 as $L_i D_i U_i$ where L and U are lower- and upper-triangular, respectively, with 1's on the diagonal, and D is invertible and diagonal. Order the bytes L_{21}, $L_{31}, \ldots$, L_{n1}, $L_{32}, \ldots$, $L_{n,n-1}$ (column order), then D^{-1} (diagonal only), then $U_{n-1,n}$, $U_{n-2,n}, \ldots, U_{1,n}$, $U_{n-2,n-1}, \ldots$, $U_{1,2}$ (U, in reversed column order). We can thus compute a product by M_1^{-1} and M_3^{-1}. Store $\mathbf{c}_1$ ahead of the M_1, and compute and tuck $\mathbf{c}_3$ away somewhere.

3.2 Testing Results with Key Scheduling

We justify, after a fashion, the claim that key scheduling [1, 39] is feasible for multivariates. We are able to fit in the keys and the program of an SFLASH scheme in 8kB ROM for a 8051-compatible micro-controller (this test was run on simulator of a development kit only) and a TTS scheme in 4kB. We can not claim our 8051 programs as well optimized, but here are the test results for TTS:

The "private key" lengths listed are really the seed maintained by the key scheduling routine. We are unable to test key generation on card with TT800 and RC4 because we somehow ran out of resources each time. Using the key setup routine for AES, the key generation time is long but barely doable. The conclusion is that *perhaps we don't need on-card key generation for such low-cost concerns*, even though there appears to be no security concerns in so doing, which can be seen from some basic testing on the public keys generated by the key-scheduled methods (Appendix A) showing "no difference."

Table 1. Key-scheduled signatures schemes on a stock 3.57MHz Intel 8032AH card

Scheme	Signature	Pr. Key	Pub.Key	setup ROM	setup	sign	sign ROM
TTS (20,28)	224 bits	1.4 kB	8.6 kB	4.2 kB	62 sec	198 ms	1.4 kB
Rijndael	224 bits	120 B	8.6 kB	6.4 kB	703 sec	454 ms	3.6 kB
RC4	224 bits	32 B	8.6 kB	N/A	N/A	334 ms	2.2 kB
TT800	224 bits	32 B	8.6 kB	N/A	N/A	310 ms	2.5 kB

4 Restructure for Faster Signing and Tiny Private Keys

Decomposing a matrix is not new. There are $[\mathbf{n}]_q! = (q^n - 1)\ (q^n - q)\ \cdots (q^n - q^{n-1})$ invertible $n \times n$ matrices over GF(q). But for generating an invertible matrix, it is common to use LU decomposition, which produces only $(q-1)^n q^{n(n-1)}$ of the non-singular matrices. To decrease storage needs, we must decompose differently here. Trying not to repeat history, we examine two such earlier attempts in Sec. 4.1.

Here we must mention the two recent other papers dealing with equivalency and normal forms of multivariates, [25] and [42]. As mentioned by [41], sparse variants like TTS are not affected much by such equivalency concerns.

4.1 Earlier Attempts

Two ideas heretofore seen in print tried replacing multiplication by a non-singular matrix by a sequence of $O(n)$ linear steps between the components of a vector.

$\mathbf{D(J-P)D}$: Both M_1 and M_3 are in form $L(J - P_\sigma)R$, where L and R are invertible and diagonal, J a matrix of all 1's, and P_σ a permutation matrix. Store L and R plus the σ's in private key.

Elementary Row Operations: [24] proposed to use both M_1 and M_3 in the form of $P_{\sigma^{-1}\rho^{-1}}\left(\prod_{i=i}^{n-1}(1+\beta_i E_{n-i+1,n-i})\right)P_\rho\left(\prod_{j=1}^{n-1}(1+\alpha_j E_{j,j+1})\right)P_\sigma D$, where n is the dimension, P_σ a permutation matrix with $\sigma = [\sigma_1, \sigma_2, \ldots]$ (list notation), D is invertible diagonal, and E_{ij} is all 0's except a 1 in the (i, j) position.

But both have security concerns. Let $\mathbf{x} = \mathsf{M}\mathbf{w}$ and $\mathsf{M} = (m_{k\ell})_{1\le k,\ell\le n}$, then

$$x_i x_j = \sum_{k=1}^{n} m_{ik}m_{jk}\; w_k^2 + \sum_{1\le k<\ell\le n} (m_{ik}m_{j\ell} + m_{i\ell}m_{jk})\; w_k w_\ell.$$

If either most of the m_{ij} are zero, or there is a tendency for $m_{ik} : m_{i\ell} = m_{jk} : m_{j\ell}$, then cross-term coefficients will tend to vanish. If M is constructed with elementary row operations as above, many entries will be zero. When both m_{ik} and m_{jk} are non-zero, the odds are good that a multiple of row i has been just added to row j or vice versa.

The matrix $L(J - P)R$ has far fewer zeroes, but most entries of $J - P_\sigma$ are 1's, hence $m_{k\ell}$ is usually $L_k R_\ell$. When neither σ_1 nor σ_2 is 1 or 2 (very likely), we have

$$x_1 = L_1(R_1w_1 + R_2w_2 + \cdots), \quad x_2 = L_2(R_1w_1 + R_2w_2 + \cdots),$$

and w_1w_2 coefficient in x_1x_2 is zero. More precisely, let $\mathsf{M}_1 = L(J-P_\sigma)R$, $\mathbf{x} = \mathsf{M}_1\mathbf{w}$. The (i, j) entry of M_1 is 0 if $j = \sigma_i$ and L_iR_j otherwise. Thus only $2n - 2$ of the $n(n - 1)/2$ cross terms are non-zero in the cross term x_ix_j.

$$x_ix_j = L_iL_j\left(\sum_{k=1}^{n} R_k^2 w_k^2 + (R_{\sigma_i}w_{\sigma_i} + R_{\sigma_j}w_{\sigma_j})\sum_{k=1}^{n} R_k w_k + R_{\sigma_i}w_{\sigma_i}R_{\sigma_j}w_{\sigma_j}\right) \quad (1)$$

It is verifiable that polynomials for $\mathbf{y}$ *in* $\mathbf{w}$ *are sparse* for cryptosystems with relatively few cross terms which leads to easier Gröbner bases computations [14].

4.2 Designing for Non-sparseness

An obvious improvement on the previous section is to use other forms of zero-one matrices that leads to fewer zeros coefficients in the quadratic polynomials.

Proposition 1. *Suppose* $\mathsf{M} = LSR$*, where* L *and* R *are diagonal and* $S = (s_{ij})$ *is a 0-1 matrix, and if* $s_{ij} = 1$ *with probability* p*, then the probability for the coefficient of* w_kw_ℓ *in the expansion of* x_ix_j *(denoted* $[w_kw_\ell](x_ix_j)$*) to be non-zero is* $\approx 2p^2(1-p^2)$.

Proof. Let entries of L and R be (resp.) L_i and R_j, then $x_i = L_i\left(\sum_{j=1}^{n} R_i s_{ij} w_i\right) + c_i$. So when $k \ne \ell$, $[w_kw_\ell](x_ix_j) = L_iL_jR_kR_l(s_{ik}s_{j\ell} + s_{i\ell}s_{jk})$. This will vanish unless exactly one of $s_{ik}s_{j\ell}$ and $s_{i\ell}s_{jk}$ is 1, or rather if and only if either $s_{ik} = s_{j\ell} = 1$ (at least one of $s_{i\ell}$ and s_{jk} is zero); or $s_{i\ell} = s_{jk} = 1$ (at least one of s_{ik} and $s_{j\ell}$ is zero).

We can make a coefficient non-zero at most half the time when $p \approx 1/\sqrt{2}$. For example, when $n = 28$ we want nineteen entries in each row and column of S to be 1 and the other nine to be 0. Elementary row operations and $D(J - P)D$ decompositions were considered partly because we want the length of the private key and the signing time to be $O(n)$, yet (as above) the number of zeros as well as the number of ones need to be $\propto n^2$. What we want is: *S should be in a form that makes* $\mathbf{x} \mapsto \mathbf{w} = \mathsf{M}^{-1}\mathbf{x}$ *doable in $O(n)$, even though* M *itself contains $\propto n^2$ of both zeros and ones.* This is where circulants come in.

4.3 Circulant Matrices

A square matrix $\mathsf{M} = (m_{ij})_{1 \le i,j \le n}$ is *circulant* if $i - k \equiv j - \ell \pmod{n}$ implies $m_{ij} = m_{k\ell}$. Circulant matrices have been studied for a long time (e.g., the monograph [7]). The S-Box of AES effectively uses a multiplication by the circulants

$$S_8 = \begin{bmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \end{bmatrix} \quad S_8^{-1} = \begin{bmatrix} 0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0\,0\,1 \\ 1\,0\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,0\,1\,0\,0 \end{bmatrix}.$$

Borrowing a page from the same book, we let $\mathsf{M}_1 = LP_\sigma SP_\rho R$, where $\sigma, \rho \in \mathcal{S}_m$ are permutations. We want an S that would let us multiply a vector by S^{-1} quickly. One obvious candidate is $S_{28} = (s_{ij})_{1 \le i,j \le 28}$, where $s_{ij} = 1$ if $j - i$ is congruent to $0 \cdots 18$ and $= 0$ otherwise. Multiplying by S_{28} is easy, but multiplying by S_{28}^{-1} is even easier, because

$$S_{28}^{-1} = (a_{i,j})_{1 \le i,j \le 28},\ a_{ij} = 1 \text{ if } i - j \equiv 0, 9, 18 \pmod{28}, \text{ else } a_{ij} = 0.$$

We would like to find an S_m for any m with similarly nice properties: $S_m^{-1}\mathbf{x}$ easily evaluated in $O(m)$, and S_m in some nice form that has the proportion of 1's as close to 0.707 as possible. Luckily, the same thing works for any $m = 3k + 1$. In fact we have

Proposition 2 (Trilinear Circulant Matrices). *In a field of characteristic 2:*

1. *Let* $S_{3k+1} = (s_{ij})$, $s_{ij} = 1$ *iff* $j - i \equiv 0 \cdots 2k \pmod{3k+1}$ *and* $= 0$ *otherwise, then* $S_{3k+1}^{-1} = (a_{ij})$ *is a 0-1 circulant with* $a_{ij} = 1$ *iff* $i - j \equiv 0, k, 2k$;
2. *Let* $S_{3k+2} = (s_{ij})$, $s_{ij} = 1$ *iff* $j - i \equiv 0 \cdots 2k \pmod{3k+2}$ *and* $= 0$ *otherwise, then* $S_{3k+2}^{-1} = (a_{ij})$ *is a 0-1 circulant with* $a_{ij} = 1$ *iff* $i - j \equiv -1, k, 2k+1$.

The proofs can be found in [45], the more complete version of this paper submitted as a research report to TWISC.

We see that our formulas are still missing the $3k$ cases. Circulants in the $DPCPD$ form seem to fit TTS (28,20) well: neither dimension is divisible by 3, and the proportion of zeros in each cross-term is $41/81$, very close to one-half. It works for SFLASH^{v2} (central map of $\mathrm{GF}(2^7)^{37} \to \mathrm{GF}(2^7)^{26}$) too, but SFLASH^{v2} takes most of its time in the central map so the time savings are scarce.

4.4 The Quirks of Circulants: Filling in the Blanks

We can summarize our problems encountered and observation made in trying to find a nice regular form for S_{3k} in the following proposition :

Proposition 3. *Let the $n \times n$ circulant $S_{n,k}$ have first row $[\underbrace{1,\ 1, \ldots, 1}_{k},\ \underbrace{0, \ldots,\ 0}_{n-k}]$, then*

1. *$S_{n,k}$ is singular if (and only if) $d = \gcd(n, k) > 1$;*
2. *For large enough n with $\gcd(n, q) = 1$, some $S_{n,k}^{-1}$ will have q lines of 1's.*
3. *If n is even, $\min_{k>1}$(# of 1's in a row of $S_{n,k}^{-1}$) is the smallest odd prime $p \not| n$.*

Here d and q are defined as odd (since we are playing in $\mathrm{GF}(2)$).

Proposition 3 implies that it is more difficult to find nice matrices for $n = 3k$ (or $n = 6k$, assuming only even dimensions). If we do not want to avoid multiples of 3, we need a circulant $S_{3k} \in \{0, 1\}^{3k\times 3k}$ with about two-thirds of its entries being 1, such that S_{3k}^{-1} has only three 1's in a row.

Proposition 4 (Trilinear Circulants of Size $3k$). *We can find circulants $\hat{C}_{k,j}$ so that $\hat{C}_{k,j}^{-1}$ exists in $\mathrm{GF}(2)$ and has $(2k - 1)$ 1's in every column. One way is to form $\hat{C}_{k,j}$ from this row:*

$$110^j10^{3k-3-j} = [1,\ 1,\ \overbrace{0, \ldots,\ 0}^{j},\ 1,\ \overbrace{0, \ldots,\ 0}^{3k-3-j}],$$

We can choose S_{3k} satisfying the requirements in Sec. 4.3 among such $\hat{C}_{k,j}^{-1}$.

Again we need to leave the proofs to the more complete Research Report [45].

4.5 Performance Evaluation

Armed with the above, for any TTS central map, we can build a TTS variant with both M_1 and M_3 in $D_L P_L C P_R D_R$ form and call it TTS LPSQR. We use only elements of the subfield $\mathrm{GF}(16)$ for the coefficients in the central map. Also, The permutations are stored packed: for dimensions under 32, we only use 5 bits to an entry, so the private key totals 289 bytes (170/2 in ϕ_2 + 20 in $\mathbf{c}_3$ + 28 in $\mathbf{c}_1$ + $(20 + 28) \times (1 + \frac{5}{8}) \times 2$

Table 2. Performance of various signature schemes on a 500MHz Pentium III

Scheme	Signature	Pub. Key	Priv. Key	KeyGen	Signing	Verifying
RSA-PSS	1024 bits	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA	326 bits	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
SFLASH	259 bits	15.4 kB	2.4 kB	1.2 sec	1.6 ms	0.28 ms
TTS (20,28)	224 bits	8.6 kB	1.4 kB	15 ms	51 μs	0.11 ms
W.LPSQR	224 bits	8.6 kB	289 B	16 ms	28 μs	0.11 ms
TTS (24,32)	256 bits	13.4 kB	1.8 kB	25 ms	67 μs	0.18 ms
W.LPSQR	256 bits	13.4 kB	455 B	27 ms	41 μs	0.18 ms

in M_1, M_3). We compare the speed of TTS, TTS LPSQR and alternatives[5] in Table 2. The tests compile with only `gcc -O3` (ver. 3.3) using plain ANSI C and no inlined assembly. For TTS, there is an expected speedup.

5 ASIC Implementation

The following represents our attempt to implement the enTTS scheme straight into an ASIC for verification purposes. This is to avoid any new vulnerabilities peculiar to structured matrices. To our best knowledge, this work appears to be the first ASIC implementation of a multivariate signature scheme in the literature, although there have been ASIC implementations of multivariate encryption schemes, e.g., NtruEncrypt [17].

In application scenarios like RFID-based security systems, typically we require a tag to authenticate itself to a reader, e.g., using the tag as a cryptographic token. In this case, only the signature functionality is needed on the tag. To meet the stringent resource constraint posed by devices like RFID tags, we only implement the signature functionality on the tags, leaving verification to the more resource-abundant (memory in particular) readers.

We first describe a hardware design that implements the plain enTTS (20,28) signature scheme without key shortening and report its synthesis result on the TSMC 0.25 μm process. The architecture of our implementation is depicted in Fig. 1 (b). It is divided into 4 portions:

1. SRAM cells: 81 bytes (8 for loop control and temporary storage, 28 for storing the signature, and 45 for solving a 9×9 linear system using Lanczos' method), taking about 9,900 gate equivalents.
2. NVRAM block: 1579 bytes (private key and look-up tables), taking about 6,400 gate equivalents.
3. Arithmetic Unit: about 800 gate equivalents, containing a set of 12 GF(16) multipliers and 2 GF(16) inverters, as well as an 8-bit integer unit for loop control.
4. Control Unit: about 3,900 gate equivalents, containing a microcoded state engine (200), microcode firmware (1,100), and an address generating unit (2,600).

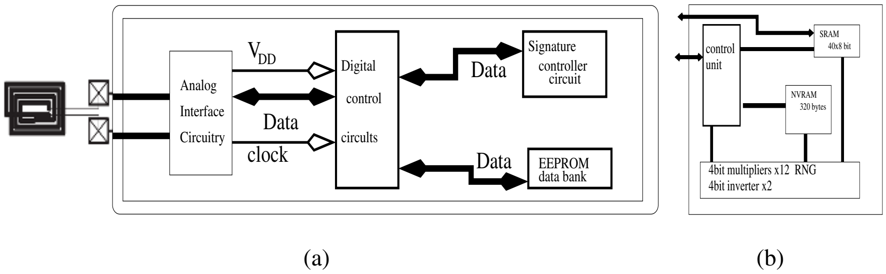


Fig. 1. Functional block diagrams for the ASIC implementation: (a) the entire RFID tag (b) inside the signature controller circuit

[5] ECC, RSA, and QUARTZ data from NESSIE website [33]. For earlier data of TTS, see [43].

The total synthesized circuit contains about 21,000 gate equivalents. Signing a message takes about 60,000 cycles, which translates to 0.6 seconds on a 100kHz clock. Most of the running time is spent in Lanczos' method. To save storage space, we do not store the matrix entries when inverting ϕ_2, so we have to generate the entries on the fly. A back-of-envelope calculation reveals that it takes about 1,000 cycles to perform a matrix-vector multiplication or to compute a bilinear form, and each iteration in the Lanczos method involves 5 such operations, adding up to a bit less than 6,000 cycles per iteration. Typically it takes 9 iterations to solve a 9×9 system, and we need to solve two such systems when inverting ϕ_2. As a result, we spend almost 90% of the time in Lanczos' method. The measured time consumed in Lanczos' method is close to 0.5 seconds per signing, fairly close to what we have estimated.

We note that it is possible to further speed up the operation by either using more SRAM or introducing parallelism. We report our attempt along the first direction. By adding 9 bytes of SRAM (totaling 54 bytes, 45 for storing a symmetric 9×9 matrix and 9 for a vector), we are able to use symmetric Gaussian elimination in place of Lanczos' method. The idea is to generate $M^T M$, solve $M^T M x = M^T w$, and then verify if $Mx = w$. This way we are able to cut the running time to about 0.16 seconds (a direct extrapolation would give 0.22, but there are some improvements we can do), less than one third the time that Lanczos' method takes. If situation permits, it is possible to halve the running time to about 0.077 seconds by adding 36 more bytes and doing a full-fledged Gaussian elimination.

At a first glance, such a strategy may not look so attractive in a resource-constrained environment such as RFID tags, particularly because SRAM cells are very expensive in terms of gate counts and power consumption. However, it will make sense from energy consumption's point of view, since cutting running time can reduce the total energy consumption and thus may achieve a higher energy efficiency. Experiments indeed validate our hypothesis. To our surprise, possibly due to the reduced switching activities in SRAM (e.g., due to less memory accesses) when doing symmetric Gaussian elimination, the power consumption of which is actually lower than that of doing Lanczos, even though the former uses more SRAM cells.

We summarize our synthesis results on the TSMC 0.25 μm process and a 100kHz clock in Table 3. Finally, with circulant (LPSQR) form of enTTS (20,28), we can go down to as low as 0.044 seconds per signing and 17,000 gates using Gaussian elimination, or 0.13 seconds and 14,000 gates using symmetrized Gaussian.

Table 3. Simulation and synthesis result on the TSMC 0.25μm process

	Lanczos		*Sym. GE*		*GE*	
Component	Gates	μA	Gates	μA	Gates	μA
SRAM	9,902	7.5	10,782	6.9	15,057	12.4
NVRAM	6,401	2.7	6,399	3.4	6,399	2.2
Arithmetic	768	4.3	768	3.5	768	4.2
Control Unit	4,207	6.2	4,009	7.5	3,689	6.6
Total	21,278	20.7	21,958	21.3	25,913	25.4

6 Concluding Remarks

NESSIE [33] did not recommend SFLASH for general use but said that it was very efficient on low-cost smart cards, where the size of the public key is not a constraint.

We think that the techniques we proposed here help with the issue of overly large keys. *This discussion applies also to other multivariate PKC schemes.* We think that multivariate schemes will be better contenders against the established schemes if they are a lot better customized and optimized.

The usual heir apparent if RSA/ECC is toppled would be NTRU. In fact NTRU-Encrypt has many good qualities. We point out that diversity is a good thing, and NTRUSign is not yet as widely implemented. So we think that there is still value to our implementations here.

We Believe That the Most Important Issue Is Still Provable Security for Multivariates. Aside from that, there are at least the following directions to explore: optimization towards smaller keys and better performance in multivariate PKCs and their security estimates. Once key-shortening techniques, e.g., in the LPSQR form or other forms, are considered safe, they can be implemented for a lot of savings in space and time, stimulating more interest towards multivariate schemes. There should still be a lot to work for in this multivariate arena.

References

1. M. Akkar, N. Courtois, R. Duteuil, and L. Goubin, *A Fast and Secure Implementation of SFLASH*, PKC'03, LNCS 2567, pp. 267–278.
2. C.-Y. Chou, Y.-H. Hu, F.-P. Lai, L.-C. Wang, and B.-Y. Yang, *Tractable Rational Map Signature*, PKC'05, LNCS 3386, pp. 244–257.
3. N. Courtois, *Generic Attacks and the Security of Quartz*, PKC'03, LNCS 2567, pp. 351–364.
4. N. Courtois, L. Goubin, W. Meier, and J. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, PKC'02, LNCS 2274, pp. 211–227.
5. N. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, EUROCRYPT 2000, LNCS 1807, pp. 392–407.
6. J. Daemen and V. Rijmen, *The Design of Rijndael, AES - the Advanced Encryption Standard.* Springer 2002.
7. P. Davis, *Circulant matrices*, John Wiley & Sons, New York-Chichester-Brisbane, 1979.
8. C. Diem, *The XL-algorithm and a conjecture from commutative algebra*, ASIACRYPT'04, LNCS 3329, pp. 338–353.
9. J. Ding, *A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation*, PKC'04, LNCS 2947, pp. 305–318.
10. J. Ding, J. Gower *et al*, *Innoculating Multivariate Schemes against Differential Attacks*, `http://eprint.iacr.org/2005/255/`.
11. J. Ding and D. Schmidt, *Rainbow, a new Digitial Multivariate Signature Scheme*, ACNS'05, LNCS 3531, pp. 164-177.
12. J. Ding and Z. Yin, *Cryptanalysis of TTS and tame-like multivariable signature schemes*, presentation, IWAP'04.
13. J.-C. Faugère, *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)*, Proceedings of ISSAC'02, pp. 75-83, ACM Press 2002.

14. J.-C. Faugère, invited talk at AES4 conference, and private communication.
15. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, *Strong Authentication for RFID Systems Using the AES Algorithm*, CHES 2004, LNCS 3156, pp. 357–370.
16. M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, Freeman and Co., 1979, p. 251.
17. G. Gaubatz, J.-P. Kaps, and B. Sunar, *Public Key Cryptography in Sensor Networks—Revisited*, 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), LNCS 3313, Heidelberg, Germany, August, 2004.
18. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, *The nesC Language: A Holistic Approach to Networked Embedded Systems*, ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI), San Diego, CA, USA, June, 2003.
19. H. Gilbert and M. Minier, *Cryptanalysis of SFLASH*, EUROCRYPT 2002, LNCS 2332, pp. 288–298.
20. W. Geiselmann, R. Steinwandt, and T. Beth, *Attacking the Affine Parts of SFLASH*, 8th International IMA Conference on Cryptography and Coding, LNCS 2260, pp. 355–359.
21. L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem*, ASIACRYPT 2000, LNCS 1976, pp. 44–57.
22. L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th Annual ACM Symposium on the Theory of Computing, (May '96) pp. 212–220.
23. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, *System Architecture Directions for Networked Sensors*, Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (November 2000), pp. 93–104.
24. Y. Hu, L. Wang, J. Chen, F. Lai, and C. Chou, *A Performance Report and Security Analysis of a fast TTM implementation*, 2003 IEEE Int'l Symp. on Information Theory, Yokohama, Japan, June 2003.
25. Y. Hu, L. Wang, F. Lai, and C. Chou, *Similar Keys of Multivariate Quadratic Public Key Cryptosystems*, CANS'05, LNCS 3810, pp. 211-222.
26. A. Joux, S. Kunz-Jacques, F. Muller, P.-M. Ricordel, *Cryptanalysis of the Tractable Rational Map Cryptosystem*, PKC'05, LNCS 3386, pp. 258–274.
27. A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, CRYPTO'99, LNCS 1592, pp. 206–222.
28. R. Lidl and H. Niederreiter, *Finite Fields*. Addison-Wesley, 1984.
29. S. Ljungkvist, in the 8051 code library `http://www.8052.com/codelib.phtm`
30. D. Malan, M. Welsh, and M. Smith, *A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography*, First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, CA, USA, October, 2004.
31. T. Matsumoto and H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, LNCS 330, pp. 419–453.
32. M. Matsumoto and T. Nishimura, *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Trans. on Modeling and Computer Sim., 8 (1998), pp. 3-30.
33. The NESSIE project homepage: `http://www.cryptonessie.org`.
34. C. Paar, *Some Remarks on Efficient Inversion in Finite Fields*, 1995 IEEE International Symposium on Information Theory, Whistler, B.C. Canada, September 1995, available from the author's website.
35. C. Paar, *A New Architechture for a Parallel Finite Field Multiplier with Low Complexity Based on Composition Fields*, Brief Contributions section of *IEEE Transactions on Computers*, vol. 45(1996), No. 7, pp. 856–861.

36. J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, CRYPTO'95, LNCS 963, pp. 248–261.
37. J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, EUROCRYPT'96, LNCS 1070, pp. 33–48.
38. J. Patarin, L. Goubin, and N. Courtois, *C^*_{-+} and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai*, ASIACRYPT'98, LNCS 1514, pp. 35–49.
39. J. Patarin, N. Courtois, and L. Goubin, *FLASH, a Fast Multivariate Signature Algorithm*, CT-RSA'01, LNCS 2020, pp. 298–307. Updated version available at `http://www.cryptonessie.org`
40. P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, Proc. 35nd Annual Symposium on Foundations of Computer Science (S. Goldwasser, ed.), IEEE Computer Society Press (1994), 124-134.
41. C. Wolf and B. Preneel, *Taxonomy of Public-Key Schemes based on the Problem of Multivariate Quadratic Equations*, `http://eprint.iacr.org/2005/077`.
42. C. Wolf and B. Preneel, *Equivalent Keys in HFE, C^*, and variations,* In Mycrypt'05, LNCS 3715, pp. 33-49, 2005.
43. B.-Y. Yang and J.-M. Chen, *Rank Attacks and Defence in Tame-Like Multivariate PKC's*, ACISP 2005, LNCS 3574, p. 518–531. Older version at E-Print Archive 2004/061.
44. B.-Y. Yang, Y.-H. Chen, and J.-M. Chen, *TTS: High-Speed Signatures on a Low-Cost Smart Card*, CHES'04, LNCS 3156, pp. 371–385.
45. B.-Y. Yang, C.-M. Cheng, B.-R. Chen, and J.-M. Chen, Technical Research Report Number 11, 2005, Taiwan Information Security Center (TWISC).

A Cryptographical Experiments

Hazards (cf. Appendix A) facing multivariate digital signature schemes include algebraic attacks, mainly XL-Gröbner-Bases [5, 8, 13], searching [4], rank attacks [21, 43], and other tailored attacks. In this section, we report our effort to make sure that our shortened keys stand up against all known attacks.

We randomly generated enTTS (with both miniature and normal instances; see [43]) and SFLASH/C^{*-} public keys (for control purposes) using both generic and LPSQR matrices, and put them through equation-solving mechanisms using Gröbner Bases, FXL simulators and rank attacks. Also small tests were run with Magma (including $\mathbf{F_4}$, the precursor to $\mathbf{F_5}$). We conducted repeated runs of tests until our test machines (with 3 GB of RAM) ran out of memory. Each test was run at least 100 times except for the maximum-sized one listed, repeated only 70 times.

Table 4. List of tests run to look for differences between generic and proposed shortened keys

Test	Test Size	enTTS	C^{*-}
FXL	Up to 14 remaining variables	no	no
MAGMA $\mathbf{F_4}$	Up to 8 eqs & vars	no	no
High Rank	Up to 16×16 matrices	no	no
Low Rank	Up to 16×16 matrices	no	no
Oil+Vinegar	Up to 12×12 matrices	no	no

The conclusion is that the shortened keys behave identically under Gröbner and XL to generic keys and do not have rank vulnerabilities. We hope to try again later with more memory and better optimized programs, perhaps by implementing Faugère's $\mathbf{F_5}$ [13] for ourselves. We must note that there is no guarantee for the security of our schemes solely based on the result of these tests, but cryptologists are still working on getting some measure of reductionist security for multivariates.

Higher Dependability and Security for Mobile Applications

Hongxia Jin

IBM Almaden Research Center, San Jose, CA, 95120
jin@us.ibm.com

Abstract. In this paper, we are concerned with the detection software faults and tampering of the mobile application as well as the mobile device theft. We want to disable mobile device cryptographically once either of these problems are detected. Basically the device needs to receive a new cryptographic key after each pre-set period of time in order to continue function. The mobile application execution integrity is checked by the authority when deciding whether or not to give out a new key. The detection can be done via a run-time result checking when the device connects to the authority. The authority can also proactively examine whether or not software tampering is happening. This paper will show approaches that each standalone can improve the dependability and security of a mobile application. We will show how these approaches can work together seamlessly to form a stronger scheme.

1 Introduction

Mobile devices, like table-PC, smartPhone and portable media center are gaining popularity. Mobile applications help one to stay in touch, get driving direction anywhere. It also allows one to check emails anywhere, synchronize one's calendar, files and life on the same schedule. Another important aspect of mobile application is entertainment. With the advent of consumer grade digital technology, content such as music and movies is no longer bound to the physical media that carries it. Mobile content becomes more and more accessible. Mobile applications allow one to listen to music and play games when one is waiting in a line or in an airplane. Digital music player like iPod is apparently a big success. Users of iPod buy and download music from website like iTune and can listen to the songs anywhere they go. Advanced mobile applications range from verifying/authorizing one's credit card, auctioning on ebay wirelessly to keeping track of inventory and customer data. In these types of applications, reliability and dependability become increasingly important. Hardware and software design and implementation errors must be minimized in order for the application to be useful.

A more serious problem is the safety and security of these applications, for example, the security of the personal information stored on the devices. By hacking the mobile devices and the applications installed on the devices, attackers can gain personal and also important business information. After success in reverse-engineering the mobile application, the attackers may also be able to get the decryption keys for the encrypted digital content out of the device and resell the content for profit. Note that digital copies are perfect copies. Furthermore, the hacker can also build a modified version of the software that gets around all the protections, and illegally redistribute this modified software to public. Security protection may come with the application to keep the critical data and confidential information safe. Digital Rights Management (DRM) and content protection software have been developed to enforce content usage. However, these technologies can only work effectively when their implementations are tamper-resistant. The development of tamper-resistant technologies, especially software tamper-resistance, has become a growth industry.

The software tamper resistance refers to the protection of software code against reverse engineering by some of its users. Many techniques have been developed to solve the opposite problem, which is to protect the host against potential hostile actions of software running on it. Relevant work in this area are Java Security [1], Software Fault Isolation [2] and proof-carrying code [3]. it should be noted that it is much more difficult to combat a malicious host than it is to combat a malicious program. Since the software runs on a hacker's machine, which has full control over its execution, the hackers can eventually completely reverse-engineer any piece of software. How can one achieve high dependability and security of mobile applications that are executed in hostile environment?

One way to protect the application program is to increase the difficulty of hacking for the attackers. Code obfuscation [4] attempts to transform a program into an equivalent one that is more difficult to manipulate and reverse engineer. A major drawback of all obfuscation approaches is that they are of necessity ad hoc. They are oftentimes not provably effective. Another technique that can provide provable protection against tampering is to encrypt programs and the program can be executed without needing to decrypt them first. Sander and Tschudin proposed *cryptographic function* [5], a way to compute with encrypted functions. They identified a class of such encrypted functions, namely polynomials and rational functions. Clearly not all programs fit into this category. Time limited blackbox security [7] creates a blackbox out of an original agent. During some time interval it is impossible for an attacker to discover relevant data or tamper the execution of the agent.

Another direction is to detect tampering. When the mobile code is sent to run on a potentially hostile host's machine, how can one make sure the returned result is correct? Furthermore, if the mobile code performs security sensitive task, how can one be confident that no tampering has been going on? Cryptographic traces have been used in [6] to show that the host duly executed the mobile agent software or some illegitimate tampering have occurred. It mainly protects the execution trace when the mobile agent roams from one site to another. We tackle the dependability and security of the mobile agent from a different angle.

In this paper, we focus on detection of tampering. For mobile applications, it is oftentimes true that the device connects to network periodically, downloading music or synchronizing with one's desktop on calendar and other files.

Indeed many of the applications mentioned earlier would be of little value if they are completely offline. With this observation, we shall present approaches that can help detect tampering. We shall present some properties that the mobile codes should have in order to facilitate the verification of the correctness of the computation and detection of any tampering occurred during the execution.

In order to detect tampering, the software usually leaves behind some kind of evidence during its execution, namely "auditing trail", and the trail can be examined later to serve as the proof of the tampering. In fact, the general concept of using an "auditing trail" to detect abnormalities has existed for a long time. Modern intrusion detection systems have attracted extensive research since Denning presented the first general intrusion detection model in [8] using auditing records. When an "auditing trail" scheme is applied to a specific detection purpose, one needs to identify what information that is needed to be put into the auditing trail, and what is the verification process that should follow to examine the trail and detect any abnormalities.

In this paper we perform run-time result verification to achieve higher reliability and dependability. If the modification of the application code causes an incorrect result for a computation, a run-time result verification approach can detect the modification. Our run-time result verification are particularly designed to support a secondary checker of a result by the authority by using the same input together with a special execution trail left behind by the primary computation in the device. We have designed the properties in the formal definition that the trail needs to satisfy in order to allow the result be checked in a provably "fool-proof" manner. The checker computation is done by the authority after the trail is returned from the device. Since the checker algorithm is only known to the authority, not anybody else, it is hard for the hacker to forge the right information to be included in the trail to fool the checker. This increases the reliability and dependability of the computations performed on the mobile application.

To further strengthen the security of the application, this paper proposes protection of the trail itself when the mobile code is executed and the trail is produced. Even when the application does not explicitly produce a computation result, our second approach adds computations into the application while trying to prevent the information in the trail from being forged. We believe the trail needs to meet the "forward security" property. Basically a one-way hash function is used to time-stamp the information that goes to the trail at any time. In this way, any information that has been recorded in the trail before the hacking success cannot be forged without being detected.

Yet another approach to improve the security of a mobile application is to detect if someone else has your mobile device and is using it. For example, you may have lost your device or someone has stolen it from you. The thief may use the credit card number stored in the device to purchase products without your knowledge. We believe it is possible to locate your mobile device. Once it is located and verified with you, the device can be disabled cryptographically by not giving out the new key, thus making the device no longer functional.

A sample motivating scenario is as follows. In a mobile system, the device needs to receive a new cryptographic key after certain (maybe even pre-set) period of time in order to continue function. The mobile application execution integrity is checked by the authority when deciding whether or not to give out a new key. If it detects any modification of the software execution, the device can be disabled cryptographically by not given a new key. We believe the three approaches shown in this paper can work together seamlessly and achieve higher reliability and security for mobile applications.

The rest of the paper is organized as follows. In section 2, we shall present our run-time result checking approach. In Section 3 we show how the information going to the trail can be protected. In Section 4, we present how the information can be used to detect mobile device theft and locate the device. We conclude in Section 6.

2 Run-Time Result Checking

A variety of techniques have been developed to help detect, and in some cases correct, errors and faults which are manifested during the execution of computer software, e.g., N-version programming [12], recovery blocks [13], algorithm-based techniques [14], checkers [15], and certification trails [11]. The errors and faults may be caused software design error, hardware transient faults, or caused by malicious tampering of the program during its execution.

A natural way of checking the correctness of of a computational result is to write another program and execute the program using the same input, then compare the two results. This is called two-version programming [12]. This approach allows for the detection of errors caused by some faults in the software in addition to those caused by transient hardware faults and utilizes both time and software redundancy. We might expect the following to hold:

1. If the two outputs are equal, then the output is correct.
2. If the two outputs are different, then an error has been detected.

But the first statement does not always hold. For example, there might be errors during both executions that cause both outputs to be incorrect, and yet equal. Thus, we must constrain the class of errors under consideration, and one natural constraint states that errors are restricted to modify only the execution of one of the programs.

If it is highly probable that only one of the programs can be wrong, then two idential outputs will indicate the result is correct and two different outputs will indicate one of the execution is wrong. Essentially in two-version programming, the second program which is used to do the result verification takes the same amount of execution time. For the same error coverage, it is desirable that the secondary verification program would take much less time and also hopefully much simpler, thus less error-prone. This in turn means the assumption that only one of the programs has errors is easier to satisfy.

With this goal in mind, our approach to run-time result verification is based on the use of program *certification trail* or *certificate*. The essence of the

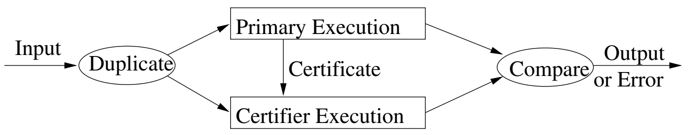


Fig. 1. Certificate-based approach

certificate-based approach,as illustrated in Figure 1, can be described as follows: we have developed methods for modifying an original program so that, with essentially negligible overhead, this modified (primary) program takes an input and not only produces a computational result, but also generates what is referred to as a *certificate*, which sometimes can be viewed as a characterization of the computational process. A secondary certifying program then takes the same input plus the certificate and either produces the same result if it is correct or otherwise indicates an error has occurred in the computation. Certificates are designed to allow the correctness of the primary program's computational result to be completely and efficiently certified in a provably "fool-proof" manner.

Note that that we must be careful in defining this method or else its error detection capability might be reduced by the introduction of data dependency between the two executions. For example, suppose the primary program execution contains an error which causes an incorrect output and an incorrect certificate to be generated. It appears possible that the execution of the certifier program might use the incorrect certificate to generate an incorrect output which matches the incorrect output produced by the primary program. The certifier must guard against an incorrect certificate "fooling" it into producing an incorrect output. The definitions we give below explicitly exclude the possibility of being fooled. In fact, if a certificate is correctly *designed* to form a certificate-based solution to a problem, the primary and certifier programs must satisfy the two properties in the definition. As shown in the definition, the second property explicitly demand that the certifier execution either generates a correct output or detects an error in the certificate-based checking, even when the primary program leaves behind an incorrect certificate during an execution.

2.1 Formal Definition of a Run-Time Result Checking Certificate

Definition 1. *Let* $\mathbf{P} : \mathbf{D} \rightarrow \mathbf{S}$ *be a problem. A solution to this problem using a certificate consists of algorithms implementing two functions* F_1 *and* F_2 *with the following domains and ranges* $F_1 : \mathbf{D} \rightarrow \mathbf{S} \times \mathbf{C}$ *and* $F_2 : \mathbf{D} \times \mathbf{C} \rightarrow \mathbf{S} \cup \{error\}$. $\mathbf{C}$ *is the set of certificates. The symbol error is chosen such that* $error \notin \mathbf{S}$ *and* $error \notin \mathbf{C}$ *and* $error \notin \mathbf{S} \times \mathbf{C}$. *The functions must satisfy the following two properties:*

(1) for all $d \in \mathbf{D}$ *there exists* $s \in \mathbf{S}$ *and there exists* $t \in \mathbf{C}$ *such that* $F_1(d) = (s,t)$ *and* $F_2(d,t) = s$ *and* $(d,s) \in \mathbf{P}$
(2) for all $d \in \mathbf{D}$ *and for all* $t \in \mathbf{C}$
either ($F_2(d,t) = s$ *and* $(d,s) \in \mathbf{P}$*) or* $F_2(d,t) = error$.

We can prove that the certificate-based method and two-version programming have similar error detection properties. That is, if errors occur in only one of the executions, they can always be detected. This is despite the fact that additional data, the certificate, is being generated and used to enable faster and simpler computation. The certificate-based approach was deliberately defined to have desirable error-detection properties. Note that when this approach is used, one must prove that the certification trail is designed to satisfy the two properties shown in the above formal definition.

However, we also note that our error model does not capture all possible different types of error behavior. For example, a hardware or software fault might cause a program to enter an ‘infinite loop’ in which case no output would be generated. A fault might cause all available computing resources to be consumed, e.g., all free memory might be allocated. The output might be too large, e.g., a very large string might be generated. These problems are not precisely modeled by the variant primed functions used in the theorems above. But note, these behaviors are relevant to each of the error detection schemes we discuss: two-version programming, certificate-based, program checkers, and others. Clearly, there must be alternative error detection facilities such as watchdog timers for ‘infinite loops’, and resource utilization monitors. In addition, if a function value lying outside its proper range or domain is encountered then an error detection flag should be raised and execution should enter an error handling routine. These auxiliary detection mechanisms are needed for each of the approaches mentioned. Our work concentrates on the types of errors which we feel are prevalent, and which are the most difficult to detect and remove, i.e., the ones that result in variant functions.

2.2 Example of a Certificate-Based Solution for a Sorting Program

It might be beneficial at this point to give a brief example of the certificate-based technique for run-time result verification of a sorting program. We will do so for the case of a sorting program. It should be understood that this example is provided for the purposes of illustration of the essence of the concept. Therefore, the discussion has been kept at a relatively high level, and its simplicity should not mislead the reader. Also, the model of the certificate described here would be slightly different for other applications.

For convenience, we assume that the sorting certifier program is given the original input vector of items to be sorted, the output of the primary sorting program, and a certificate. The certifier must verify if the output is indeed correct. The first step performed by the certifier program is to test whether the output is monotonically increasing. Note, that this test is not sufficient to prove the correctness of the output. The certifier must also check that the output vector is a permutation of the input vector. To do this, the certifier utilizes the certificate. Assuming that the primary program operated in an error free manner, the certificate C is the permutation of indices that when applied to the input vector produces the output vector. More precisely, let I be the input vector, O the output vector, and C the certificate. The certifier program computes a vector P, such that $P[i] = I[C[i]]$. The certifier program compares P to O. If they are the same, the certifier answers that the output is correct; otherwise, the certifier program answers that the output is incorrect. It should be clear that the certifier program is easily implemented to run in linear time, provided that array accesses are assumed to take only constant time.

As an example, let I, O, and C be as follows:

$$I = [12.3, 7.2, 17.4, 16.1, 5.2]$$

$$O = [5.2, 7.2, 12.3, 15.1, 17.4]$$

$$C = [5, 2, 1, 4, 3]$$

Then the vector P constructed would be

$$P = [5.2, 7.2, 12.3, 16.1, 17.4]$$

Thus the check that $O = P$ would fail. Also notice that there is no vector C would allow the test $O = P$ to succeed in this case. It is beyond the scope of the discussion here that the sorting certificate described above adheres to the definition and theorem of Section 2.1. Nevertheless, the example does convey some sense of the concept.

Of course, the certification trail is different for different applications. We believe this approach can offer higher reliability and dependability on some typical applications on mobile devices. For example, mobile devices are oftentimes used to find driving directions. The application attempts to find the nearest or closest way to a destination. A wireless auction application may need to find the best bid for the user. Below we will show how we can certify computations results for search-type applications.

2.3 Certifying Search-Type Computations

We have conducted a preliminary investigation of search-type problems which are amenable to 'branch-and-bound' strategies. This powerful strategy is widely used; for example, it is possible to solve the classic Traveling Salesman problem using a branch-and-bound strategy [16]. We believe that it is possible to use the certificate-based method to efficiently assure the correctness of many computations which are solved using a branch-and-bound strategy.

Consider a typical branch-and-bound search attempts to find an optimal solution which minimizes or maximizes some objective function. The search conducted by a branch-and-bound procedure can be represented implicitly by a tree called the *search tree* or a directed-acyclic gragh (DAG). Each node in the search tree is associated with a sub-problem. Initially, the search tree of the branch-and-bound procedure contains a single root node which corresponds to the original problem. During a branch-and-bound search a problem is split into multiple sub-problems and these sub-problems are further split into more sub-problems. The lower-bound and/or upper-bound calculations are used to determine if the search from a given sub-problem can be halted.

For a branch-and-bound computation, the certificate generated by the primary consists of a tree or DAG which represents the search conducted during the execution of the primary. The tree or DAG is 'annotated' with important information about the bounding computations performed by the primary. The certifier must traverse the tree or DAG and use the annotations to determine if the primary computation manifested any errors and to certify whether the answer determined by the primary is correct.

Three goals are pursued when the certificate-based method is applied. First, the primary should not be substantially slower than the original computation which does not generate a certificate. Second, the certifier computation should be substantially faster than the primary; and third, the certifier should be simpler than the primary (hence providing diversity). Our preliminary work indicates that very often these goals can be achieved.

Solving and Certifying TSP. As an example of our ideas, we have applied our technique to an algorithm solving TSP problem [16] in which there are N cities and for each pair of cities there is a known cost of travel between the cities. A traveling salesman wishes to find a minimum cost tour which visits each city exactly once and which returns to the starting city. When this algorithm performs action (1), the splitting of a problem into sub-problems, it first chooses an edge by using a heuristic procedure. The problem is then split into exactly two sub-problems. In one sub-problem, the edge must be in the solution tour; and in the other sub-problem, the edge must not be in the solution tour. Clearly, the solution to the problem being split must also be the solution to one of the two sub-problems. Action (2), solving a sub-problem, is performed when two edges remain to be added to yield a tour. Action (3), the bounding calculation, is performed as described below. A sample search tree is shown in Figure 3.

The cost associated with each pair of cities can be represented in a cost matrix. A TSP tour is a simple cycle which includes each city exactly once. Thus, the TSP tour contains exactly one element from each row and each column of the cost matrix A. When a constant q is subtracted from each element in some row of A, or in some column of A, the cost associated with any possible tour is reduced by exactly q. It follows that, the relative costs of any two tours remains unchanged, and thus the optimal tour remains unchanged with respect to the original cost matrix and the modified cost matrix. If such subtractions are performed upon rows and columns such that at least one element in each row and each column becomes zero, while every other element remains nonnegative, the total amount subtracted from the rows and columns is a lower bound on the cost of any TSP solution with respect to the original matrix. We call this process **reduction**.

As illustrated in Figure 2, the certificate for TSP algorithm contains the optimal TSP tour and the search tree. The information annotated in each tree node is the splitting edge together with the accumulated subtractions for rows and columns. One of the main routines used in certifying the correctness of the TSP tour attempts to check the correctness of the lower bounds calculated

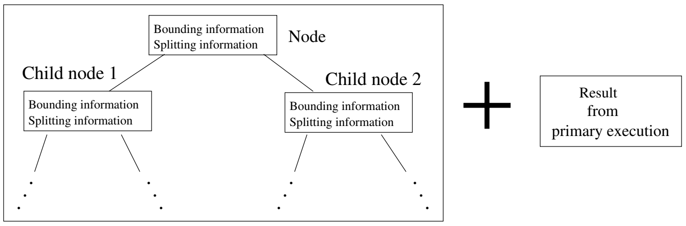


Fig. 2. The certificate for a generic branch-and-bound algorithm. It consists of the obtained result and the search tree generated from the primary computation. The tree is annotated with information about bounding computations and splitting operations.

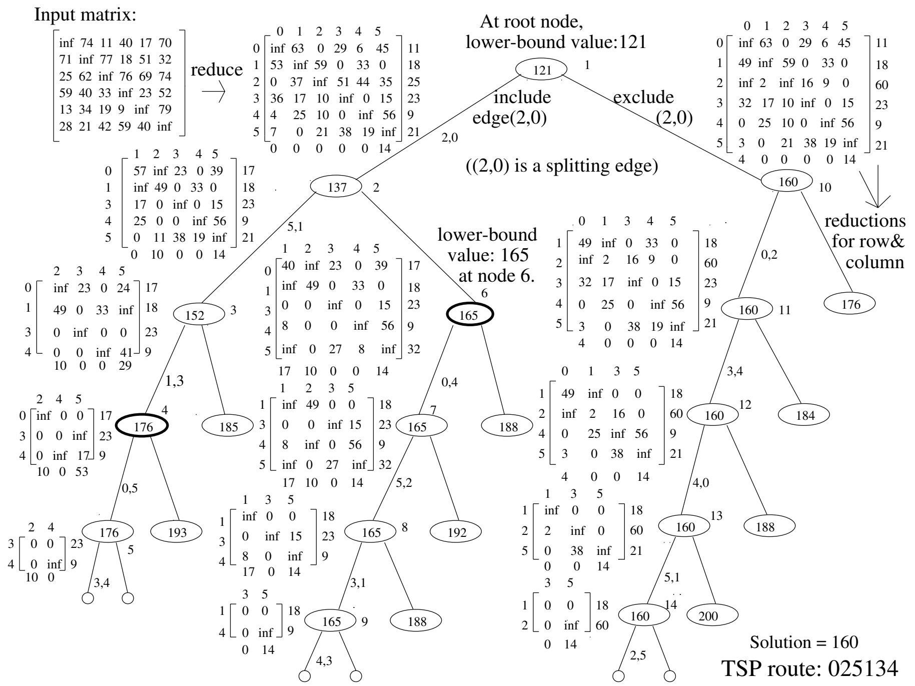

Input matrix:
reduce
At root node, lower-bound value:121
include edge(2,0)
exclude (2,0)
((2,0) is a splitting edge)
lower-bound value: 165 at node 6.
reductions for row& column
Solution = 160
TSP route: 025134

Fig. 3. The entire search tree for a six node TSP example; Note that the number marked inside the node is the lower-bound value for the subproblem associated with that node. The edge written on each left branch is the splitting edge: the left branch includes it, and the right branch excludes it. The numbers marked below and to the right of each matrix are the row reductions and column reductions respectively.

by **reduction** in the primary algorithm. In Figure 3, these lower bounds are shown in each tree node. Checking these lower bounds can be done efficiently if the accumulated totals of the subtracted values for each row and column are available. In Figure 3, these accumulated totals are shown below and to the right of each matrix. Specifically, the totals for rows and columns that have not been deleted during the search are shown, and we refer to these rows and columns as *active*. To certify the correctness of a lower bound, the program uses knowledge of which edges have been included and excluded from a sub-problem, together with the accumulated subtractions for rows and columns.

The general idea of function **lowerbnd_check** is that a matrix entry in the original matrix should be greater than or equal to the sum of the corresponding row accumulator and column accumulator when it is an entry in an active row and column. Further, the lower bound for a sub-problem should equal the sum of all the row and column accumulators plus the cost of the edges included in the sub-problem.

The certifier algorithm traverses the search tree in a depth-first manner visiting the left node first. When the search comes to a node whose lower bound value is equal to or greater than the *supposed_sol_value*, which indicates the search can halt at this node, the certifier checks the correctness of the lower bound computations by performing a **lowerbnd_check**. While traversing the search tree, the certifier performs other checks such as **splitting_validity_check** to make sure the tree is well formed. While we cannot show the TSP primary and certifier algorithms here, we can prove in the following theorems that the set of checks are adequate for showing that the obtained TSP solution is correct. If one of the checks fails, then an error message is generated.

Theorem 1. *Function* **reduction** *and* **lowerbnd_check** *together form a correct certificate-based solution to the TSP lower-bound computation.*

Theorem 2. *The branch_and_bound* **TSP_primary** *algorithm and the branch_and_bound* **TSP_certifier** *algorithm together form a correct certificates-based solution to the TSP problem.*

2.4 Experimental Results

We have performed experiments to see how much performance overhead incurs when certification trail approach is applied to the TSP problem. The timing data was gathered using a Sun SPARCstation ELC running Sun OS 4.0. The system was run as a stand alone machine during the timing experiments. All timing data listed is in seconds.

The entries used in the table are as follows. The *Basic Algorithm* column gives the execution time of the original algorithm for producing the output without generating the certification trail. The *Primary Execution* column gives the execution time of the algorithm for producing the output with the additional overhead of generating the certification trail. The *Certifier Execution* column gives the execution time of the algorithm in producing the output using the certification trail. The *Speedup* column gives the ratio of the execution times of the basic algorithm and the certifier execution.

Examination of the data indicates that the overhead in generating the certification trail is rather small, around 1% of the running time of the basic algorithm which does not generate a certification trail. The table also shows how much time the certifier can save when using the certification trail technique. It is notable

Table 1.

Size	Basic Algorithm	Primary Exec (Also Generates Trail)	Certifier Exec (Uses Trail)	Speedup
20	0.72	0.73	0.04	18.00
30	7.77	7.87	0.30	25.90
40	105.61	105.87	3.51	30.09
50	1636.36	1645.62	41.16	39.76
60	3321.00	3328.44	73.55	45.15

that the speedup also increases with the size of the input. This is a desirable property when certifying computationally intensive tasks.

3 Protecting the Trail

We believe the run-time result verification improves the reliability and dependability of a mobile application. As mentioned earlier, for certificate-based approach, the checker program is only known to the authority. This makes it hard for the attackers to forge the right information that goes to the trail. However, for mobile applications that demands higher security protection and also when one faces more determined attackers, it is desirable to protect the trail. After all, the application executes on the attacker's machine and the attacker has full control over the execution. A determined attacker can always "eventually" understand the logging process. But we assume before the attacker completely understands the logging scheme, he/she tumbles first. When he/she tumbles, it causes some of the information that is logged into the trail to be wrong.

Taking advantage of the mobile application's periodic network connection, we hope we can detect the tampering before he/she succeeds the hacking. Our goal is to make sure the incorrect information recorded in the trail caused by the tampering cannot be forged back to become correct without being detected. When the attacker succeeds in the tampering, he/she can forge any future information, but he cannot go backwards. In other words, it has to satisfy the "forward security" property [9]. Our idea to achieve this is to add another dimension on the information in the trail. That dimension is time. There is a one-way function that keeps evolving once the program starts to execute. When any value is logged into the trail, it relates to the current one-way function value [10].

There are two ways to perform one-way logging. In the first approach, a one-way function $f1$ is used to evolve keys independent of the actual value v_i that goes to the trail. $f1(k_i) = k_{i+1}$. When it needs to store a value v_i to the trail, another $f2$ function uses the key on value v_i to generate the actual entry in the trail. For example, $f2$ might be the encryption of the key k_i with value v_i. Because the key evolves using a one-way function, at time t when a hacker succeeds with the tampering and gets the current key k_i, he still has no way to know the previous keys $k_0 \ldots k_{i-1}$. Therefore, he cannot go back and forge any entries already recorded in the trail. When the entries are transmitted back to authority, the authority can repeat the $f1$ computation and decrypt the values in the trail. Then using all the values in the trail to verify the result as described in the previous section.

Another way of logging the information into the trail more securely is as follows. When it needs to store a value v_i into the trail, it can just do it. But at sametime, a one-way function f uses both current key k_i and the current value v_i to generate a new key k_{i+1}.

$$f(k_i, v_i) = k_{i+1}$$

The trail, together with the last obtained key k_n, are transmitted back to authority. When software is hacked causing some of the v_i values wrong, if the hackers forge the values back to the correct value, the key evolution calculated by the authority will be different than the k_n value returned. If the hacker does not correct the v_i values themselves, then the incorrect values in the trail will fail the result verification that can be done as described in the previous section.

Compared to the previous logging approach, this approach has the property that the actual trail is in the clear. This avoids the authority's need to decrypt the trail before performing result verification.

Note that this online tampering detection approach can be used alone [10] where the information that goes to the trail can be integrity check values at chosen places throughout the application. However, when this approach is combined with the run-time result checking, it improves the strength of both approaches.

4 Locating Your Mobile Device

The ubiquity of the mobile device also increases the possibility for the owner to lose it or the device be stolen. We believe it is possible to find the lost device and prevent it from being used illegitimately by someone without your authorization.

Taking advantage of the periodic connection made by the device, it is possible to make the device send back some identifying information back to authority when it is connected again, for example, the serial number and IP address. The authority can always email or phone the owner to verify the loss. The email address and phone numbers are obtained during the one-time user registration time.

5 A Complete Scheme

As described in earlier sections, the approaches we proposed here can be used standalone. Run-time result verification can be used to mainly improve the reliability and dependability of the application and detect potential software design and implementation error. By inserting integrity checks at various places in the program, the online detection can potentially detect the ongoing tampering even before the tampering finally succeeds. If the authority receives report from customer about the lost of his/her device, or the lost/stolen device sends back serial number and IP address, it can also refuse to give out new keys and make it nonfunctional.

However, as a complete scheme, these approaches can seamlessly work together and form a system with higher dependability and security. For example, when the certificate-based approach is used for run-time result verification, it decides what is the information during execution that needs to be written into the trail. It is decided based on the formal certificate definition. The program is thus modified accordingly to write the information into the trail. In addition to that, integrity checks can be inserted at critical places on all execution paths. The information that goes to the trail and the integrity check values are protected using the one-way function. When connected online, the trail and maybe the integrity check values are transmitted back to the authority. The information in the trail may need to be decrypted first before it can be used to verify the result. Note that the information going to the trail could be the intermediate execution result of the running application. Remember each device is given a different initial key. When the authority receives back the trail, it knows from which device the trail comes from. In other words, some self-identifying information must be included in the trail. If someone picks up your lost device or steals it, when the device connects again and sends back the trail. The authority can simply trace back the message route that the trail is sent from, it can detect the location of the device. As we can see, in this complete scheme, no separate information needs to send back in order to locate a lost/stolen device.

Imagine a scenario that can use all the approaches together to achieve higher dependability and security for a mobile application, the mobile device needs to receive a new cryptographic key after certain period of time in order to continue function. The mobile application execution integrity is checked by the authority when deciding whether or not to give out a new key. If it detects any modification of the software execution or the tampering of the software, or the device has been lost/stolen, the device can be disabled cryptographically by not given a new key.

6 Conclusion

As stated in [17], it is impossible to prevent the malicious or faulty host from tampering the mobile code. In this paper we attempt to detect tampering to achieve higher security and dependability of a mobile application. We have presented the properties that the mobile codes need to have in order to perform dependable and secure computation when they are executed on a potentially hostile machine. Our scheme uses certificate-based approach for run-time result verification. When the application runs, it not only generates a result as the output, it also leaves behind a trail. When the application connects to the network, the computational result together with the trail is transmitted back to the authority. With the help of the trail, the authority can verify the correctness of the result and the execution faster as well as in a simpler way thus less error-prone. We have shown the properties that the trail needs to satisfy in order to facilitate an efficient verification of the computational result. In general it needs to keep the checker algorithm secret and make it hard for the hacker to mimic the right information that should be included in the certificate and prevent the checker from being fooled. To strengthen the approach and add more protection during execution, this scheme also protects the trail using a one-way function that keeps evolving immediately after the execution starts. With this protection, the information going to the trail cannot be forged without being detected. Integrity checks can be inserted at critical points in the program and the integrity check results can be transmitted back to authority, the authority can detect tampering. Even though these methods can each be used standalone, we believe when these methods are seamlessly integrated together, they offer a stronger scheme that offers higher dependability and security for a mobile application. In fact, no extra information needs to be sent back and tracing the message route of the trail can easily identify the approximate location of a lost/stolen device.

Note that there is no general way of design certificate, it is still application specific. As future work, we would like to expand the approach to other computational problems and applications as well as find a general theory and methodologies for certificate design for any application. When used for mobile application, the trail needs to be transmitted via network, it is also of our interest to design space-efficient trails. Furthermore, it is of our interest to investigate the automated support of the certificate generating process and we like to build tools to automatically insert integrity checks into the application. We like to continue investigate the secure storage of the one-way key value and explore different ways to make use of the information that goes to the trail as well as additional information that may be useful in detecting other security breaches.

References

1. Steven Fritzinger and Marianne Mueller. *Java Security*, 1996
2. Robert Wahbe, Steve Lucco, T.E. Anderson, and Susan Graham. *Efficient software-based fault isolation*, In proceedings of the ACM SIGCOMM 96 symposium, 1996.
3. George Necula. *Proof Carrying Code*. In proceedings of the Twenty Fourth Annual Symposium on Principles of Programming Languages, 1997.
4. Don Libes. *Obfuscated C and other mysteries*. Wiley, 1993.
5. Tomas Sander and Christian F. Tschudin, *Protecting mobile agents against malicious hosts*, Mobile agents and Security, p44-60, 1998
6. Giovanni Vigna. *Cryptographic traces for mobile agents*. Mobile Agents and Security, LNCS 1419, pp. 137-153, 1998.
7. F. Hohl, *time limited blackbox security: protecting mobile agents from malicous hosts*, Mobile agents and security, Springer-Verlag, pp 92-113, 1998.
8. Dorothy Denning, *An intrusion detection model*, IEEE Transactions on Sofwtare engineering, vol. SE-13, No.2, Feb, 1987, pp222-232.
9. A. J. Menezes, P.C.van Oorschot, and S.A.Vanstone, *Handbook of applied cryptograghy*. CRC press, 1997, ISBN 0-8493-8523-7.
10. H.Jin and J.Lotspiech, "Proactive software tampering detection", in proceeding of ISC 2003.
11. Sullivan, G.F., and Masson, G.M., "Using certification trails to achieve software fault tolerance," *Digest of the 1990 Fault Tolerant Computing Symposium*, pp. 423-431, IEEE Computer Society Press, 1990.

12. Avizienis, A., “The N-version approach to fault tolerant software,” *IEEE Trans. on Software Engineering*, vol. 11, pp. 1491-1501, Dec., 1985.
13. Randell, B., “System structure for software fault tolerance,” *IEEE Trans. on Software Engineering*, vol. 1, pp. 220-232, June, 1975.
14. Huang, K.-H., and Abraham, J., “Algorithm-based fault tolerance for matrix operations,” *IEEE Trans. on Computers*, pp. 518-529, vol. C-33, June, 1984.
15. Blum, M., and Kannan, S., “Designing programs that check their work”, *Proceedings of the 1989 ACM Symposium on Theory of Computing*, pp. 86-97, ACM Press, 1989.
16. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B., “The Traveling Salesman Problem”, John Wiley and Sons Ltd., 1985.
17. David Chess, Benjamin Grosof, Colin Harrison, David Levine, Colin Paris and Gene Tsudik, ”Itinerant Agents for Mobile Computing”, Technical report, IBM T.J.Watson Research Center, 1995

Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

Alexander Becher, Zinaida Benenson, and Maximillian Dornseif

RWTH Aachen, Department of Computer Science,
52056 Aachen, Germany
alex@cyathus.de, zina@i4.informatik.rwth-aachen.de,
dornseif@informatik.uni-mannheim.de

Abstract. Most security protocols for wireless sensor networks (WSN) assume that the adversary can gain full control over a sensor node through direct physical access (node capture attack). But so far the amount of effort an attacker has to undertake in a node capture attack is unknown. In our project we evaluate different physical attacks against sensor node hardware. Detailed knowledge about the effort needed for physical attacks allows to fine tune security protocols in WSNs so they provide optimal protection at minimal cost.

1 Introduction

Wireless sensor networks (WSN) consist of a large amount of *sensor nodes*, which are small low-cost wireless computing devices equipped with different sensors. They gather and process environmental data like temperature, humidity, light conditions, seismic activities. Sensor nodes are also sometimes called *motes* because of their small size and the envisioned deployment pattern: They are supposed to be spread over a large geographic area, organize themselves into an ad hoc network, and operate unattended for months or even years. Many intended applications, such as military surveillance, but also structural health monitoring (detecting damage in buildings, bridges, aircrafts), supply chain management, or building protection, have high security requirements. For an overview of security issues in sensor networks, see [23, 24].

One of possible attacks on WSNs is called *node capture*. This describes a scenario where an adversary can gain full control over some sensor nodes through direct physical access. As sensor nodes operate unattended and cannot be made tamper proof because they should be as cheap as possible, this scenario is more likely than in most other computing environments. This type of attack is fundamentally different from gaining control over a sensor node remotely through some software bug. As all sensors are can be assumed to run the same software, finding an appropriate bug would allow the adversary to control the whole sensor network. In contrast, a node capture attack can only be mounted on a small portion of a sufficiently large network.

Depending on the WSN architecture, node capture attacks can have significant impact. Thus, most existing routing mechanisms for WSNs can be substantially influenced even through capture of a minute portion of the network [14].

In the TinySec mechanism [13], which enables secure and authenticated communication between the sensor nodes by means of a network-wide shared master key, capture of a single sensor node suffices to give the adversary unrestricted access to the WSN.

Most current security mechanisms for WSNs take node capture into account. It is usually assumed that node capture is "easy". Thus, some security mechanisms are verified with respect to being able to resist capture of 100 and more sensor nodes out of 10,000 [6, 12]. However, to the best of our knowledge, nobody ever tried to determine the actual cost and effort needed to attack currently available sensor nodes.

Thus we set out to verify the assumption that node capture is easy. The contributions of this work are as follows:

1. We developed a design space for *physical attacks* on sensor nodes. These are all attacks that require having direct physical access to the sensor nodes.
2. We found out that node capture attacks which give the adversary full control over a sensor node are not so easy as usually assumed in the literature. They require expert knowledge, costly equipment and other resources, and, most important, removal of nodes from the network for a non-trivial amount of time.[1]
3. We conclude that removal of a sensor node from the deployment area can be noticed by its neighbors, as well as by the sensor node itself. Using appropriate measures (see Section 6), the affected sensor node can be timely excluded from the network.
4. Therefore, we looked into possibilities to attack unattended sensor nodes *in the field*, without disruption of the regular node operation. We actually found some attacks (and also countermeasures) which are described in Section 5, and evaluated them in experimental setups.

Roadmap. First, in Section 2, we give an overview of previous work on physical and tampering attacks on embedded systems in general and how they relate to sensor networks. We also give pointers to existing work concerning forms of attacks against WSNs which we do not consider here. In Section 3, we describe current sensor node hardware. Section 4 gives our design space for attacking sensor nodes, and Section 5 presents some examples for attacks and defenses. In Section 6, recommendations on WSN design resistant against node capture attacks are given, and ongoing and future work is presented.

2 Related Work

Vogt et al. [33] consider the case of *recovering* nodes that have been the target of a successful attack and have fallen under the control of an attacker. They describe

[1] However, this applies only if the WSN designers take some basic precautions which are well known in the world of embedded systems, such as disabling the JTAG interface (see Section 5.1), or protecting the bootstrap loader password (see Section 5.2).

an intrusion detection and recovery mechanism which can be implemented in software on certain types of microcontrollers.

Bugs in the software running on the sensor nodes or on the base stations give rise to attacks which can be easily automated and can be mounted on a very large number of nodes in a very short amount of time. Possible countermeasures include a heterogenous network design and standard methods from software engineering [22, 11, 32, 16].

Physical attacks on embedded systems, that is, on microcontrollers and smart cards, has been intensively studied before [2, 26, 1, 25]. Skorobogatov describes in depth tampering attacks on microcontrollers, and classifies them in the three categories of *invasive*, *semi-invasive*, and *non-invasive* attacks [25]. Invasive attacks are those which require access to a chip's internals, and they typically need expensive equipment used in semiconductor manufacturing and testing, as well as a preparation of the chip before the attack can begin. Semi-invasive attacks require much cheaper equipment and less time than the invasive attacks, while non-invasive attacks are the easiest.

All of these attacks, including the so-called low-cost attacks, if applied to sensor nodes, would require that they be removed from the deployment area and taken to a laboratory. Even if in some cases, the laboratory could be moved into the deployment area in a vehicle, all attacks would require at least disruption of the regular node operation. Most of the invasive and many of the semi-invasive attacks also require disassembly or physical destruction of the sensor nodes.

Skorobogatov also lists several possible classification schemes, including U. S. government standards, both for attackers, according to their capabilities, and for defenses, according to their abilities to resist attacks from a certain adversary class.

The existing literature on physical attacks usually assumes that an attacker can gain unsupervised access to the system to be attacked for an extended period of time. This is a sensible assumption for systems such as pay-per-view TV cards, pre-paid electricity tokens, or GSM SIM cards. Attacks which may take days or even weeks to complete present a real threat to the security of these systems.

In wireless sensor networks, however, regular communication with neighboring nodes is usually part of normal network operation. Continuous absence of a node can therefore be considered an unusual condition that can be noticed by its neighbors. This makes time a very important factor in evaluating attacks against sensor nodes, as the system might be able to detect such attacks while they are in progress and respond to them in real-time. One of our aims has been to determine the exact amount of time needed to carry out various attacks. Based on these figures, the frequency with which neighbors should be checked can be adapted to the desired level of security and the anticipated threat model.

Finally, the focus of previous work has been mostly on attacking the components themselves as opposed to the entire products. Attacks on the circuit-board level have been deliberately excluded from many works, although they are recognized to be security-relevant in some cases. We did not exclude such attacks from our investigation since our focus was on the security of the entire node and not only of its individual components.

3 Current Sensor Node Hardware

Currently available sensor nodes typically consist of embedded hardware with low power consumption, and low computing power. A typical sensor node contains some sensors (light, temperature, acceleration etc.), a radio chipset for wireless communication, an EEPROM chip for logging sensor data, a node-to-host communication interface (typically a serial port), and a microcontroller which contains some amount of flash memory for program storage and RAM for program execution. Power is provided by batteries.

Figure 1 shows a general schematic for the hardware of current sensor nodes, while Figure 2 shows photographs of some concrete models available today.

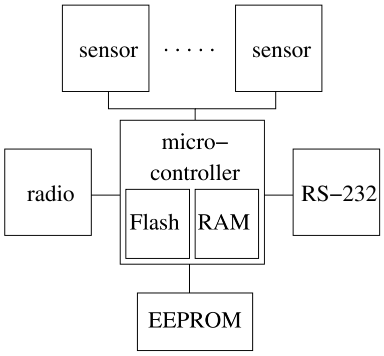


Fig. 1. General schematic of sensor node hardware

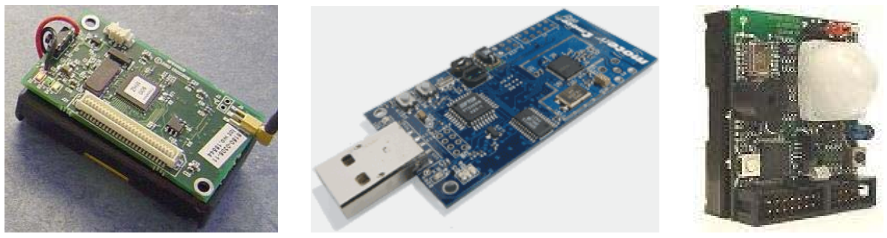


Fig. 2. Current sensor node hardware: Mica 2 by Crossbow, Berkeley (from [15]); Tmote sky by moteiv, Berkeley (from [20]); and Embedded Sensor Board by ScatterWeb, Berlin (from [5])

Typical choices for the microcontroller are the 8 bit Atmel ATmega 128 or the 16 bit Texas Instruments MSP430 family, with the amount of RAM varying between 2 kB and 10 kB and flash memory ranging from 48 kB to 128 kB. External EEPROM memory can be as small as 8 kB or as large as 1 MB. The speed of radio communications is in the order of 100 $^{\text{kbit}}/_{\text{s}}$.

The most interesting part for an attacker will no doubt be the microcontroller, as control over this component means complete control over the operation of the node. However, the other parts might be of interest as well in certain attack scenarios. Our classification scheme in Section 4 takes this into account, and Section 5 presents some examples for attacks on other components.

3.1 Mica2

The Crossbow Mica2 nodes [9,10] (fig. 2, left) use the 8 bit Atmel ATmega 128 microcontroller [4] with 4 kB RAM and 128 kB integrated flash memory, the Atmel AT45DB041B 4 Mbit flash memory chip [3], and the Chipcon CC1000 radio communications chipset [7] with a maximum data rate of 76.8 kbit/s. Programming is done via the Atmel's serial programming interface by placing the node in a special interface board and connecting it to an RS-232 serial port on the host.

3.2 Telos

The Telos motes [19,20] (fig. 2, center) by Moteiv utilize the Texas Instruments MSP430 F1611 microcontroller [31,30], providing 10 kB of RAM and 48 kB flash memory. The EEPROM used is the 8 Mbit ST Microelectronics M25P80 [27], and the radio chipset is the Chipcon CC2420 [8], whose maximum data rate is 250 kbit/s. Programming is performed by connecting to the USB interface and writing memory with the help of the MSP430 bootloader [28]. A JTAG interface is available as an alternative programming method and can also be used for debugging.

3.3 ESB

The Embedded Sensor Boards [5] (fig. 2, right) from ScatterWeb GmbH are built around the Texas Instruments MSP430 F149 microcontroller [31,29] featuring 2 kB of RAM and 60 kB flash memory, the Microchip 24LC64 [17] 64 kbit EEPROM, and the RFM TR1001 radio chipset [18] with a maximum data rate of 19.2 kbit/s. Programming is done either through a JTAG interface or over-the-air using a gateway.

4 Design Space for Physical Attacks on Sensor Nodes

4.1 Physical Attacks vs. Tampering

The majority of previous work on hardware security has focused on the security of single components. As different hardware platforms for wireless sensor networks are very similar to each other, we need not restrict ourselves to tampering attacks on the components themselves, but can include e. g. attacks on the circuit board level. This allows us to conduct a more detailed security analysis of the entire hardware instead of simply analyzing the security of every part on its own.

The term “tampering” is well accepted in the research community to designate attacks on components that involve modification of the internal structure of a single chip. At the same time, there are also conceivable attacks on sensor node hardware which do not really fit this accepted usage. In order to avoid this terminology problem, we call those attacks that we regard *physical attacks* and use this term to refer to all attacks requiring direct physical access to the sensor node.

4.2 Design Space

We propose the following classification scheme for physical attacks. This design space takes our previous considerations into account that sensor nodes are more or less in permanent contact with each other. This means that long interruptions of regular operation can be noticed and acted upon. Therefore, attacks which result in a long interruption (e. g. because the node has to be physically removed from the network and taken to a distant laboratory) are not as dangerous as those which can be carried out *in the field*. The main focus of our project has been on this class of attacks as well as on possible countermeasures to be employed by a sensor network.

The intention of our design space is to enable system designers to evaluate the feasibility and severity of a concrete attack under a concrete adversary model against their concrete system.

The two main categories that we use for classifying physical attacks are (1) the degree of control over the sensor node the attacker gains; and (2) the time span during which regular operation of a node is interrupted. Figure 3 illustrates this design space and classifies example attacks from Section 5 according to its criteria.

According to the degree of control the attacker gains, we classify the attacks into following categories, which are listed here in order of decreasing severity:

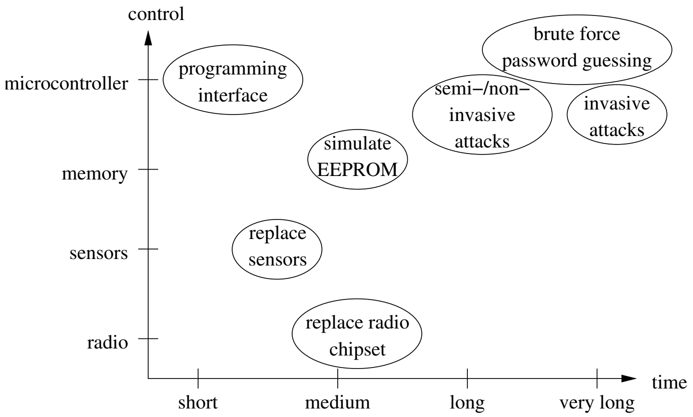


Fig. 3. Design space for physical attacks on sensor nodes

1. The attacker gains complete read/write access to the microcontroller. This gives the attacker the ability to analyze the program, learn secret key material, and change the program to his own needs.
2. The attacker learns at least some of the contents of the memory of the node, either the RAM on the microcontroller, its internal flash memory, or the external flash memory. This may give the attacker, e. g., cryptographic keys of the node.
3. The attacker is able to influence sensor readings.
4. The attacker can control the radio function of the node, including the ability to read, modify, delete, and create radio messages without, however, having access to the program or the memory of the sensor node.

We then classifiy the attacks according to the time during which the node cannot carry out its normal operation. In our design space, we use the following four possibilities:

1. Short attacks of less than five minutes. Attacks in this class mostly consist of creating plug-in connections and making a few data transfers over these.
2. Medium duration attacks of less than thirty minutes. Most attacks which take this amount of time require some mechanical work, for instance (de-) soldering.
3. Long attacks of less than a day. This might involve a non-invasive or semi-invasive attack on the microcontroller, e. g., a power glitch attack where the timing has to be exactly right to succeed, or erasing the security protection bits by UV light.
4. Very long attacks which take longer than a day. These are usually invasive attacks on the electronic components with associated high equipment cost.

Three other, less important from our point of view, properties associated with an attack can be illustrated as follows:

– The *cost* an attacker is required to spend in terms of equipment to carry out an attack successfully: This can range from extremely cheap, where only a soldering iron and some cables are required, to prohibitively high, where top-of-the-line semiconductor test equipment is needed.
– The *skill and knowledge* that an attacker has to possess for a successful attack: Some attacks might be carried out by a kid after proper instruction, while others might require extensive knowledge of the particular application of the sensor network, or a person trained in the use of special equipment. This property can also be modeled as *cost*.
– The *traces* left behind by the attack: If after the attack the node is left in the same state as before the attack, including unaltered memory contents, then this is harder to notice than an attack which causes physical destruction of the node.

5 Examples of In-the-Field Attacks and Countermeasures

Our project set out to actually implement attacks on wireless sensor networks. Our aim has been to gather some real-world data about currently known, proposed, and previously unidentified attacks. First, we wanted to measure the effort needed to carry out existing attacks, and the effort needed to devise new attacks. Second, we intended to evaluate the effectiveness of various existing security mechanisms and to come up with new ones. In the rest of this section, we will describe some of the attacks that we have investigated, together with possible countermeasures, and classify them according to the criteria described above.

5.1 Attacks Via JTAG

The IEEE 1149.1 JTAG standard is designed to assist electronics engineers in testing their equipment during the development phase. Among other things, it can be used in current equipment for on-chip debugging, including single-stepping through code, and for reading and writing memory.

A JTAG Test Access Port (TAP) is present on both the Atmel and Texas Instruments microcontrollers used on the sensor nodes described above. All sensor nodes examined by us have a JTAG connector on their circuit board allowing easy access to the microcontroller's TAP. While the capabilities offered by JTAG are convenient for the application developer, it is clear that an attacker must not be provided with the same possibilities. Therefore it is necessary to disable access to the microcontroller's internals via JTAG before fielding the finished product.

The MSP430 has a security fuse which can be irreversibly blown (as described in the data sheet) to disable the entire JTAG test circuitry in the microcontroller. Further access to the MSP430's memory is then only possible by using the Bootstrap Loader described in Section 5.2. The ATmega128 requires the programmer to set the appropriate fuses and lock bits, which effectively disable all memory access via JTAG or any other interface from the outside.

If JTAG access is left enabled, an attacker equipped with an appropriate adapter cable and a portable computer is capable of taking complete control over the sensor node. Even if there is no JTAG connector provided on the circuit board, attackers can still get access to the JTAG ports by directly connecting to the right pins on the microcontroller which can be looked up in the datasheet. Typical data rates for JTAG access are 1–2 kB/s, so reading or writing 64 kB of data takes between 30 and 70 s. However, there are specialized programming devices on the market which can attain much higher data rates. One such device claims to be able to program 60 kB of memory in a mere 3.4 s.

5.2 Attacks Via the Bootstrap Loader

On the Telos nodes, the canonical way of programming the microcontroller is by talking to the Texas Instruments specific bootstrap loader (BSL) through the USB interface. The bootstrap loader [28] is a piece of software contained in the ROM of the MSP430 series of microcontrollers that enables reading and writing the microcontroller's memory independently of both the JTAG access and the program currently stored on the microcontroller.

The BSL requires the user to transmit a password before carrying out any interesting operation. Without this password, the allowed operations are essentially "transmit password" and "mass erase", i.e. erasing all memory on the microcontroller.

The BSL password has a size of $16 \cdot 16$ bit and consists of the flash memory content at addresses 0xFFE0 to 0xFFFF. This means in particular that, immediately after a mass erase operation, the password consists of 32 bytes containing the value 0xFF. The memory area used for the password is the same that is used for the interrupt vector table, i.e. the BSL password is actually identical to the interrupt vector table. The interrupt vector table, however, is usually determined by the compiler and not by the user, although Texas Instruments documents describe the password as user-settable.

Finding out the password may be quite time-consuming for an attacker. However, such an investment of time may be justified if a network of nodes all having an identical password is to be attacked. Therefore, an evaluation of the possibility to guess the password follows.

Brute Force. As the password is composed of interrupt vectors, certain restrictions apply to the values of the individual bytes. This section examines the expected size of the key space and estimates the expected duration of a brute force attack on the password.

Initially, the key space has a size of $16 \cdot 16\,\text{bit} = 256\,\text{bit}$. Assuming a typical compiler (mspgcc 3.2 [21] was tested), the following restrictions apply:

– All code addresses must be aligned on a 16 bit word boundary, so the least significant bit of every interrupt vector is 0. This leaves us with a key space of $16 \cdot 15\,\text{bit} = 240\,\text{bit}$.
– The reset vector, which is one of the interrupt vectors, is fixed and points to the start of the flash memory, reducing the key space to $15 \cdot 15\,\text{bit} = 225\,\text{bit}$.
– Interrupt vectors which are not used by the program are initialized to the same fixed address, containing simply the instruction `reti` (return from interrupt). As a worst case assumption, even the most basic program will still use at least four interrupts, and therefore have a key space of at least $4 \cdot 15\,\text{bit} = 60\,\text{bit}$.
– Code is placed by the compiler in a contiguous area of memory starting at the lowest flash memory address. Under the assumption that the program is very small and uses only $2\,\text{kB} = 2^{11}\,\text{B}$ of memory, we are left with a key space of a mere $4 \cdot 10\,\text{bit} = 40\,\text{bit}$.

We conclude that the size of the key space for every BSL password is at least 40 bit.

A possible brute force attack can be performed by connecting a computer to the serial port (the USB port, in the case of Telos nodes) and consecutively trying passwords. This can be done by executing a modified version of the `msp430-bsl` [21] program that is normally used for communicating with the BSL.

The rate at which passwords can be guessed was measured to be approximately 12 passwords per second when the serial port was set to 9600 baud. However, the MSP430 F1611 used on the Telos nodes is capable of a line speed of 38,400 baud, and at this speed, approximately 31 passwords can be tried per second. Finally, the BSL program normally waits for an acknowledgment from the microcontroller after each message sent over the serial line. If this wait is not performed, the speed of password guessing rises to 83 passwords per second.

The maximum speed of password guessing in practice can therefore be assumed to be 2^7 passwords per second. This is quite close to the theoretical limit of $38,400\,{}^{\text{bit}}/_{\text{s}} \cdot (256\,{}^{\text{bit}}/_{\text{pw}})^{-1} = 150\,{}^{\text{pw}}/_{\text{s}}$.

Recalling that the key space has a size of at least 40 bit, we can now conclude that a brute force attack can be expected to succeed on the average after $2^{40-7-1}\,\text{s} = 2^{32}\,\text{s} \approx 128\,\text{a}$. As 128 years is well beyond the expected life time of current sensor nodes, a brute force attack can be assumed to be impractical.

Knowledge of the Program. One consequence of the fact that the password is equal to the interrupt vector table is that anyone in possession of an object file of the program stored on a sensor node also possesses the password. Worse, even someone who only has the source code of the program still can get the password if he has the same compiler as the developer, since he can use this compiler to produce an image from the source code identical to the one on the deployed nodes.

The secret key in the current TinySec implementation, for example, is contained in the image but does not influence the interrupt vector table. If TinySec were ported to Telos motes, the source code and the compiler used would be sufficient information for an attacker to extract the secret key material. The same holds for any kind of cryptographic mechanism where the key material does not influence the interrupt vectors.

Another way of exploiting the identity of the password and the interrupt vector table is to take one node away from the network and attack the microcontroller on this node with classic invasive or semi-invasive methods. The absence of the node from the network will probably be noticed by the surrounding nodes and its key(s) will be revoked (see also Section 6). However, once the attacker succeeds with her long-term attack and learns the BSL password of the one node, it is trivial for her to attack all of the other nodes in the field if they all have the same BSL password.

If an attacker knows the BSL password, reading or writing the whole flash memory takes only about 25 s.

In order to avoid these forms of attack, we propose a technique called *interrupt vector randomization.* We have designed and implemented a program called `rand_int` that operates on a program image in Intel hex format produced by the compiler. It can be used to preprocess an image before installation on a node. Our tool reads the old interrupt vector table from the image file and replaces it in the following way:

1. While reading the image, all used memory areas are remembered and output unchanged.
2. When the interrupt vector table is read, it is not output directly.
3. Instead, for every original address in the interrupt table, an unconditional branch instruction to this address is generated.

4. Then an unused memory area is randomly chosen, the branch instruction is placed there, and that memory region is marked as used.
5. Finally, the entry in the interrupt table is replaced by the address where the branch instruction was placed.

The resulting image file then leads to a BSL password that can neither be guessed nor derived from the source code, effectively preventing third parties from reading the sensor node's memory even if they have access to the source code of the application used on the node, or if they have attacked a different node and learned its BSL password. Care should be taken to erase the image file with randomized interrupt vectors after programming the node.

This approach could also be extended for over-the-air reprogramming. Instead of performing the randomization process on the developer's host and sending an individual image to every node in the network, an identical image could be broadcast to all nodes. The randomization process would then have to take place on every individual node after it receives the new image.

5.3 Attacking the External Flash

Some applications might want to store valuable data on the external EEPROM. For example, the Deluge implementation of network reprogramming in TinyOS stores program images received over the radio there. If these images contain secret key material, an attacker might be interested in reading or writing the external memory.

Probably the simplest form of attack is eavesdropping on the conductor wires connecting the external memory chip to the microcontroller. Using a suitable logic analyzer makes it easy for the attacker to read all data that are being transferred to and from the external EEPROM while she is listening. If a method were found to make the microcontroller read the entire external memory, the attacker would learn all memory contents. This kind of attack could be going on unnoticed for extended periods of time, as it does not influence normal operation of the sensor node.

A more sophisticated attack would connect a second microcontroller to the I/O pins of the flash chip. If the attacker is lucky, the mote microcontroller will not access the data bus while the attack is in progress, and the attack will be completely unnoticed. If the attacker is skillful, she can sever the direct connection between the mote microcontroller and the flash chip, and then connect the two to her own microcontroller. The attacker could then simulate the external memory to the mote, making everything appear unsuspicious.

Of course, instead of using her own chip, the attacker could simply do a "mass erase" of the mote's microcontroller and put her own program on it to read the external memory contents. This operation is even possible without knowledge of the BSL password. While this causes "destruction" of the node from the network's point of view, in many scenarios this might not matter to the attacker.

The exact amount of time required for the attacks proposed above remains to be determined. It should be noted that some of the attacks outlined above require a high initial investment in terms of equipment and development effort. A possible countermeasure could be checking the presence of the external flash in regular intervals, putting a limit on the time the attacker is allowed to disconnect the microcontroller from the external flash.

5.4 Sensors

Sensor nodes rely on their sensors for information about the real world, so the ability to forge or suppress sensor data can be classified as an attack. For instance, a surveillance system might be tricked into thinking that the situation is normal while the attacker passes unnoticed through the area under surveillance.

Replacing sensors on the different types of nodes varies in difficulty between child's play and serious electrical engineering, mostly depending on the type of connection between the microcontroller circuit board and the sensors. A pluggable connection—as present on the Mica2 motes—requires an attacker to spend only a few moments of mechanical work. If, on the other hand, the sensors are integrated into the printed circuit board design, replacing them involves tampering with the conductor wires, cutting them, and soldering new connections. The amount of time required for this will vary with the skill of the attacker, but it can be assumed to be in the order of minutes.

5.5 Radio

Finally, the ability to control all radio communications of a node might be of interest to an attacker. At the moment, we do not have any concrete attack which involves replacing the radio chipset, but we believe that it can still prove useful in some attack.

6 Conclusion

We systematically investigated physical attacks on current sensor node hardware, paying special attention to attacks which can be executed directly in the deployment area, without interruption of the regular node operation. We found out that most serious attacks, which result in full control over a sensor node (*node capture*), require absence of a node in the network for a substantial amount of time. We also found simple countermeasures for some of the most serious attacks,

Thus, in order to design a WSN secure against node capture attacks, the following steps should be applied:

- take standard precautions for protecting microcontrollers from unauthorized access;
- choose a hardware platform appropriate for the desired security level, and keep up-to-date with new developments in embedded systems security;
- monitor sensor nodes for periods of long inactivity;
- allow for revocation of the authentication tokens of suspicious nodes.

Standard precautions for protecting microcontrollers from unauthorized access, such as disabling the JTAG interface, or protecting the bootstrap loader password, are an absolute prerequisite for a secure sensor network. We developed a method of protecting the bootstrap loader password by randomization of the interrupt vector table. This allows the developers to make source code of their products public without fearing that their WSN can be taken over by everybody who compiles the source code using the same compiler, thus obtaining the same interrupt vector table, and therefore, the same BSL password.

As security is a process, not a product, system designers should keep up-to-date with the developments in attacks on embedded systems. The security of important systems should be constantly re-evaluated to take new discoveries into account, as newly found attack methods on microcontrollers or previously unknown vulnerabilities might make a previously impossible low-cost attack in the field possible.

The level of security required from the application should also be kept in mind when choosing hardware. In some cases it might make sense to build additional protection, such as a secure housing, around a partially vulnerable microcontroller.

Finally, the removal of a sensor node from the deployment area can be noticed by its neighbors using, e. g., heartbeat messages or topology change notifications, as well as by the sensor node itself using, e. g., acceleration sensors. Appropriate measures can then be taken by the network as well as by the node itself. The network might consider a node that has been removed as "captured" and revoke its authorization tokens or initiate recovery when this node returns to the network. The node itself might react to a suspected physical attack by erasing all confidential material stored on it.

Mechanisms should be developed that allow a sensor node which has been absent for too long from the network to be revoked by its neighbors. This is our future work. Note that depending on the WSN design, local revocation could be insufficient. For example, if an attacker removes a single sensor node from the network and successfully extracts the node's cryptographic keys, the attacker would be able to *clone* nodes, to populate the network with new sensor nodes which all use the cryptographic keys of the captured sensor node. Thus, a WSN should also be protected from node cloning.

References

1. Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, Inc., 2001.
2. Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136, London, UK, 1998. Springer-Verlag.
3. Atmel Corp. AT45DB041B datasheet. Atmel document no. 3443, available at `http://www.atmel.com/dyn/resources/prod_documents/doc3443.pdf`.
4. Atmel Corp. ATmega128 datasheet. Atmel document no. 2467, available at `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

5. FU Berlin. ScatterWeb Embedded Sensor Board. Online at http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/esb/.
6. Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, May 2003.
7. Chipcon AS. CC1000 datasheet. Available at http://www.chipcon.com/files/CC1000_Data_Sheet_2_3.pdf.
8. Chipcon AS. CC2420 datasheet. Available at http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf.
9. Crossbow, Inc. MICA2 data sheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.
10. Crossbow, Inc. MPR, MIB user's manual. Available at http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf.
11. Mark G. Graff and Kenneth R. van Wyk. *Secure Coding: Principles and Practices.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.
12. Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52. ACM Press, 2004.
13. Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.
14. Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols*, September 2003.
15. Geoff Martin. An evaluation of ad-hoc routing protocols for wireless sensor networks. Master's thesis, University of Newcastle upon Tyne, May 2004.
16. Gary McGraw and John Viega. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison-Wesley, September 2001.
17. Microchip Technology. 24AA64/24LC64 datasheet. Available at http://ww1.microchip.com/downloads/en/DeviceDoc/21189K.pdf.
18. RF Monolithics. TR1001 datasheet. Available at http://www.rfm.com/products/data/tr1001.pdf.
19. moteiv Corp. Telos revision B datasheet. Available at http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf.
20. moteiv Corp. Tmote Sky datasheet. Available at http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf.
21. http://mspgcc.sourceforge.net/.
22. Holger Peine. Rules of thumb for secure software engineering. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 702–703, New York, NY, USA, 2005. ACM Press.
23. Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.
24. Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6), December 2004.
25. Sergei P. Skorobogatov. Semi-invasive attacks - a new approach to hardware security analysis. Technical report, University of Cambridge, Computer Laboratory, April 2005. Technical Report UCAM-CL-TR-630.
26. Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 2–12, London, UK, 2003. Springer-Verlag.

27. STMicroelectronics. M25P80 datasheet. Available at `http://www.st.com/stonline/products/literature/ds/8495.pdf`.
28. Texas Instruments. Features of the MSP430 bootstrap loader (rev. B). TI Application note SLAA089B, available at `http://www-s.ti.com/sc/psheets/slaa089b/slaa089b.pdf`.
29. Texas Instruments. MSP430 F149 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f149.pdf`.
30. Texas Instruments. MSP430 F1611 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f1611.pdf`.
31. Texas Instruments. MSP430x1xx family: User's guide. TI Application note SLAU049E, available at `http://www-s.ti.com/sc/psheets/slau049e/slau049e.pdf`.
32. John Viega, Matt Messier, and Gene Spafford. *Secure Programming Cookbook for C and C++*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.
33. Harald Vogt, Matthias Ringwald, and Mario Strasser. Intrusion detection and failure recovery in sensor nodes. In *Tagungsband INFORMATIK 2005, Workshop Proceedings*, LNCS, Heidelberg, Germany, September 2005. Springer-Verlag.

The Brave New World of Ambient Intelligence: An Analysis of Scenarios Regarding Privacy, Identity and Security Issues

Michael Friedewald[1], Elena Vildjiounaite[2], Yves Punie[3], and David Wright[4]

[1] Fraunhofer Institute for Systems and Innovation Research, Breslauer Straße 48, 76139 Karlsruhe, Germany
[2] Technical Research Centre of Finland, VTT Electronics, Kaitoväylä 1, 90571 Oulu, Finland
[3] European Commission/DG JRC, Institute for Prospective Technological Studies, Edificio EXPO, C/Inca Garcilaso, 41092 Seville, Spain
[4] Trilateral Research & Consulting, 12 Tybenham Road, London, SW19 3LA, United Kingdom

Abstract. The success of Ambient Intelligence (AmI) will depend on how secure it can be made, how privacy and other rights of individuals can be protected and how individuals can come to trust the intelligent world that surrounds them and through which they move. This contribution presents an analysis of ambient intelligence scenarios, particularly in regard to AmI's impacts on and implications for individual privacy. The analysis draws on our review of more than 70 AmI projects, principally in Europe. It notes the visions as well as the specifics of typical AmI scenarios. Several conclusions can be drawn from the analysis, not least of which is that most AmI scenarios depict a rather too sunny view of our technological future. Finally, reference is made to the SWAMI project (Safeguards in a World of Ambient Intelligence) which, inter alia, has constructed "dark" scenarios, as we term them, to show how things can go wrong in AmI and where safeguards are needed.

1 Introduction

The term Ambient Intelligence (AmI) was coined by Emile Aarts of Philips and taken up by the Advisory Group to the European Community's Information Society Technology Program (ISTAG) as the convergence of ubiquitous computing, ubiquitous communication, and interfaces adapting to the user. The concept of AmI depicts a vision of the future information society, where the emphasis is on greater user-friendliness, more efficient services support, user empowerment, and support for human interactions. People are surrounded by intelligent intuitive interfaces that are embedded in all kinds of objects and an environment that is capable of recognising and responding to the presence of different individuals in a seamless, unobtrusive, and often invisible way [1, 2].

Privacy, identity, security and trust are central key issues in ambient intelligence visions and have been identified as such from their earliest inception [3].

Many in the research and development community clearly recognise the inherent challenge that an invisible, intuitive and pervasive system of networked computers holds for current social norms and values concerning privacy and surveillance.

The inherent privacy challenge from ambient intelligence stems from two innovations necessary to its success: the enhanced ability to collect data on people's everyday interactions (in multiple modalities and over large spans of time and space) and an enhanced ability to quickly search large databases of that collected data, creating greater possibilities for personal profiling, and other forms of data mining [4]. One leading researcher in the field has identified a set of generic privacy concerns that ambient intelligence will very likely raise for users [5]:

– A pervasive network of interconnected devices and communications will mean that the sheer quantity of personal information in circulation will increase greatly;
– The introduction of perceptual and biometric interfaces for certain applications will transform the qualitative nature of personal information in circulation;
– In order to offer personalised services, ambient intelligence will require the tracking and collection of significant portions of users' everyday activities.

It has to be noted that many of the foreseen concerns unfold as the technology develops. As of today, they seem to be still far away and some visions sound even like science fiction. However it is important to deal with them early on in order to shape the future in a desirable way. To understand the directions of thinking of AmI and its inherent threats, a screening of more than 70 R&D projects and roadmaps, many of which have developed scenarios, was undertaken – most of them from EU-funded research projects such as the "Disappearing Computer Initiative" [6, 7, 8, 1, 9, 10, 11, 12, 13, 14, 15, to name the most important].

In the analysed papers the AmI vision of everyday life is a mixture of many diverse applications ranging from relatively easy-to-realise prototypes to scenarios in the more distant future taken from roadmaps. We have clustered the many aspects of our future everyday lives in the following application domains: home, work, health, shopping and mobility.

Before detailing our analytic approach to deconstructing AmI scenarios, we present in the following section the views of several researchers on privacy and its aspects.

2 Aspects of Privacy

Privacy is generally considered to be an indispensable ingredient for democratic societies. This is because it is seen to foster the plurality of ideas and critical debate necessary in such societies. Bohn et al. [4] refer to the work of Lessig [16] and argue that there are different dimensions related to privacy and new technologies, in particular Information and Communication Technology (ICT). The following aspects of privacy are identified:

Privacy as empowerment. Privacy has an informational aspect. People should have the power to control the publication and distribution of personal information.

Privacy as utility. From the viewpoint of the person involved, privacy can be seen as a utility providing more or less effective protection against nuisances such as unsolicited phone calls or e-mails. This view follows a definition of privacy as "the right to be left alone".

Privacy as dignity. Dignity not only entails being free from unsubstantiated suspicion (for example, being the target of a wire tap, where the intrusion is usually not directly perceived as a disturbance), but also focuses on the equilibrium of information available between two people.

Privacy as a regulating agent. Privacy laws and moral norms can be seen as a tool for keeping checks and balances on the powers of a decision-making elite.

Furthermore, Bohn et al. [4] say that people perceive their privacy being invaded when borders are crossed. The following borders are identified:

Natural borders. Observable physical borders, such as walls and doors, clothing, darkness, sealed letters and phone conversations can represent a natural border against the true feelings of a person.

Social borders. Expectations with regard to confidentiality in certain social groups, such as family members, doctors, and lawyers include, for example, the expectation that your colleagues do not read personal fax messages addressed to you.

Spatial or temporal borders. Most people expect that parts of their lives can exist in isolation from other parts, both temporally and spatially. For example, a previous wild adolescent phase should not have a lasting influence on an adult's life, nor should an evening with friends in a bar influence his coexistence with work colleagues.

Borders due to ephemeral or transitory effects. This describes what is best known as a "fleeting moment," a spontaneous utterance or action that we hope will soon be forgotten. Seeing audio or video recordings of such events subsequently, or observing someone sifting through our trash, would violate our expectations of being able to have information simply pass away unnoticed or forgotten.

Other authors develop these aspects of privacy further. For example, Nissenbaum [17] presents a model of informational privacy in terms of contextual integrity, namely, that determining privacy threats needs to take into account the nature of a situation or context: what is appropriate in one context can be a violation of privacy in another context.

Singer [18] argues that privacy is not only about disclosure of dangerous information or disturbing a person at a wrong time or with information on a wrong topic. For example, personalisation may seem to be beneficial, since it reduces the amount of useless and annoying advertisements. However, this may be not so innocent or beneficial in the long term because advertisers view people as bundles of desires to buy more, and may result in diminishing people's capacities of reasoned choice and thoughtful action.

3 Deconstructing AmI Scenarios

3.1 Analytical Framework

Producing scenarios is generally a way to present in a concise form the most visible research activities in a certain application domain. AmI application scenarios can be found in at least three different forms.

First, there are *elaborated scenarios* (screenplays) with actors and their activities, with many details and a well-defined storyline. These scenarios can be either purely conceptual, theoretical visions of a future (good examples are the ISTAG scenarios [1]) or scenarios developed by projects aiming at presentation of a project goal.

Second, there are *application scenarios*, which can be found in research publications. These scenarios usually concentrate on a certain functionality of a system prototype described in the publication, and the storyline in these scenarios is detailed only in parts, which describe the system functionality.

The third and most common types are not scenarios in the strict sense. They do not present concise storylines but rather describe situations and/or drivers or trends that may give rise to relevant AmI applications. (See, for instance, Michahelles [19] on how technology may help to save lives of avalanche victims when skiing.) Such descriptions often suggest interesting application areas, which one may not find in more elaborated scenarios of the first or second types. However, it would be a mistake to miss such approaches because they show existing prototypes.

For the decomposition and analysis of visions and scenarios from numerous sources, the following dimensions were explored in the texts [20]:

- the personality of the main *actors*, because social and privacy issues depend on the scenario target, e.g., people with disabilities may be willing to exchange some privacy to get more support; and small children do not care about privacy yet;
- the *environment* where the scenario takes place, because people have different expectations about privacy in different environments, e.g., in their own home where people are less willing to accept the same behavioural restrictions as in public places;
- the *activity* described in the scenario, because activity is an important part of a personal context;
- the *information flow* in the scenario, because many privacy threats are associated with disclosure of information;
- the *control level* of the envisaged AmI system, because it leads to higher dependability on AmI and because it raises a lot of questions about legal responsibility for actions taken by the technical system. This also affects humans' acceptance of AmI;
- *enabling technology* because many privacy threats are associated with the actual system implementation.

3.2 The Envisioned AmI User Population

As there are different target groups in the scenarios, actors in the scenarios are described by their age group, health group, profession, social status and attitude towards technology. This dimension is important because social, legal and privacy issues depend on scenario actors, e.g., people with severe disabilities may be willing to exchange some privacy to get more support, and small children don't care much about privacy. On the other hand, politicians usually care much more about personal data than average people.

Most of the analysed scenarios feature ordinary working people, and it is assumed that most people, including the elderly, will have embraced AmI. With the exception of scenarios describing support for shopping and everyday activities for elderly people (in most of the scenarios, they live alone), support for such basic everyday activities as shopping, watching TV etc. are often described as an individual activity of a healthy, well-off adult.

AmI that is focused on the individual can create problems in family relations. For example, in scenarios describing how an intelligent TV is able to select only the channels and programs that are really interesting for the user (e.g., by measuring the user's physiological signals), it is rarely mentioned that there can be several family members with conflicting interests. The ITEA scenario "the Rousseaus' holiday" is one of a few exceptions in this sense [9, pp. 134ff.]. In scenarios describing how a user is woken by cheerful music, the user is either assumed to be sleeping alone or that all family members wake up at the same time and by the same music, which is not always desirable [15]. Similarly, shopping scenarios often neglect the fact that shopping is not an individual but rather a social activity, where family members often have very different responsibilities and decision rights. It is seldom analysed that children may have the right to put certain things on a shopping list, but that parents need to be notified and given the right to veto this decision [6]. The roles of children in the scenarios are usually restricted to playing games, being checked by parents and receiving reminders to do homework or to collect right things. They are rarely presented as active human beings with their own responsibilities.

Significant effort is devoted to supporting communications between humans. Communications between family members, relatives, friends, colleagues, and strangers can be asynchronous (messaging) or synchronous (video communications), at home, at work (both on non-working and working issues) and while moving. However, many scenarios describing communications between adults and children present them in such a way that parents activate the video link or notification service in order to check what their children are doing, and it is not clear whether the children have rights to avoid being observed by parents at any time. Children are described as activating communications with adults mainly when adults are travelling. This neglects the important role that youngsters have always played in the adoption of new media (e.g. SMS).

Health care scenarios are different from scenarios in other domains in the sense that they are targeted at people with chronic diseases, health risks, elderly people and people with disabilities. However, the general pattern is that a person's problems or disabilities are described as if there was only an AmI solution to help them. Most scenarios imply that AmI itself works excellently and does not create problems for people. Another general rule is that scenarios describing smart environments (whether it is a smart shop or city-wide ambient intelligence) and basic human activities (such as shopping or going to work) assume that all people have accepted and can afford the new technologies.

3.3 Activities and Environments in Ambient Intelligence

The typical scenarios that have been developed over the last years envision application of ambient intelligence for work, learning, ordinary everyday activities like household-related activities, changing places, leisure and hobby, social, health maintenance, emergency. This dimension is important because information flow is closely linked to the activity, and main privacy threats are related to information flow.

Home, being the most private place for people, needs to be designed carefully because the home atmosphere is important for personal happiness and development [21]. If a spy wants sensitive personal information about somebody, the best way to get it is to observe that person at home. Many financial and private affairs are discussed or dealt with from home, personal vulnerabilities, strengths and health problems can be seen easily, and it is difficult to say what information about someone can not be found in the home environment. Second, people perceive their homes as a place where they can be free from intrusion, relax and think in peace, i.e., "to be left alone". As Nissenbaum said, this privacy aspect is very important for personal development [17].

The *work* domain has three noteworthy aspects: first, people spend a lot of time working, but they are less free to choose their working environment than their home environment. If organisations choose to violate personal privacy in some way, workers can either accept it or try to find a better employer. In any case, if workers feel their privacy is violated, they can feel humiliated and depressed, and it is an open question how much an organisation will benefit or lose from close surveillance of its employees. Second, people can not avoid dealing with some personal matters during working hours, e.g., making appointments to see a doctor or a teacher of their children; talking to a mechanic about repairs to their cars; communicating with family members; reserving a table at a restaurant and so on. It is difficult to avoid doing some personal things because working hours are more or less the same everywhere and children may need parental permission or advice during working hours. Thus, it follows that intellectual property is not the only thing that should be protected in a working environment. Personal affairs also need to be protected. Third, our analysis of research projects targeted at developing AmI at work has shown that visions of the future working environment are already being implemented in some companies, hospitals and research institutes capable of investing more money than ordinary people in their homes. Thus, we can expect to work in a smart environment sooner than to live in a smart home. Consequently, safeguards of privacy at work should be developed as soon as possible.

The *health* domain has two aspects: on the one hand, health care determines the life and death of people, and fast access to a person's health information (e.g., allergies and chronic diseases) can be very important in case of emergency. On the other hand, health information is highly sensitive. People may be unwilling to reveal their health problems even to close relatives, let alone to work superiors or insurance companies. Thus, it is important (but maybe not so easy) to build AmI applications in the health domain so that emergency workers and doctors can access personal information whenever needed, but nobody else can do so without authorisation.

Ambient intelligence applications in *shopping and commerce* aim at creating a user-friendly, efficient and distributed service support to the customer, such as managing the search for and selection of merchandisers by the customer, and handling order and payment processes.

In the applications of AmI for *travel and mobility* (called the "nomadic domain" by one influential scenario [9]) will have the same facilities and services as those in the home and at work, but while people are at different places temporarily or on the move (e.g., on the road). The mobility domain has two aspects: first, people are not free to control the environment where they move - governments require passports, visas, driving licences, etc; transportation companies have rules too, e.g., where to put the luggage and what can or can not be transported. AmI technologies are already becoming present in the mobility domain in the form of biometric passports, supported by governmental financing, which will soon become obligatory in Europe. Second, people travel both for work and for their own pleasure. Travel is an important feature of life today. This means that privacy protection in the mobility domain needs to be developed urgently, otherwise travellers will be left with the choice either of accepting threats to their privacy or ceasing to travel (and for those who travel for work ceasing travel is simply impossible).

All these activities do not necessarily take place in one environment, in fact it is part of the AmI vision that activities can take place at virtually any place. The environments found in the scenarios include: urban, long-distance travelling, countryside and nature. This dimension is important because in different environments different technologies are needed, and because it shows where changes can appear earlier. The dimension is also important because people have different expectations about privacy in different environments, e.g. in their own home and in the nature people are less willing to accept same behaviour restrictions as in public places.

3.4 Information Flow in AmI Applications

In most scenarios, the AmI system recognises people, either for the purpose of access control or for personalisation. In many scenarios, it is left open how exactly personal identification is performed, but there are indications that people have either an "identity token" that can be read by the system or biometrics are used. Both possibilities have identity theft risks associated with them [22].

Scenarios which require high security (like immigration control, protection of professional secrets, or access to health data) and which mention biometric sensors do not usually describe which biometrics are used. However, it seems probable that highly reliable biometrics, such as iris scanning or fingerprint reading, will be used in high-security applications, and theft of highly reliable biometrics data is very dangerous. It is worth noting that identity information is always stored somewhere (in a personal device or in a central database or both) and it is always exchanged during the authentication process. The presence of identity information in multiple locations increases a risk of identity theft, particularly when one takes into account the fact that currently information stored in personal devices is poorly protected.

Another very popular element of scenarios is the presence of information about a person's or object's location and/or destination. Most often, it is processed locally, in the user device or in the car, but it can also be transmitted to a service provider. Tracking of workers' location and location of work-related objects is also seen as a common functionality of AmI, and in such scenarios, workers' locations are not always processed locally, but are sent to a central server instead.

One more common scenario element is the automatic payment of road tolls and other fees, as well as automatic payment for purchases. This implies that credit card details are stored in a personal device and transmitted during the payment process. Other personal financial data, such as available budget, is also known to AmI systems in work and home environments.

Intimate and sensitive data such as health information is also often stored either locally on a smart card or another personal/wearable device – which can get lost or stolen – or in a central database which may not be sufficiently secured and, even if it is, data can be misappropriated by malicious employees. Moreover, since health information is needed in more than one place, a large amount of data transmission is associated with health applications. This includes the regular transmission of new data from sensors to central databases, but also extensive ad hoc communication. First, personal/wearable devices have to communicate with systems in the physician's surgery and in the hospital. During this ad hoc communication, the most sensitive information (identity, health history, etc.) is exchanged. Second, mobile emergency systems use ad hoc communication with third party nodes as relays for data transmission [15]; the communication devices of other cars and a gas station are used to transmit an emergency call that includes sensitive information about the identity of the injured person. It is also worth noting that health data can be acquired not only during health monitoring, but also during evaluation of a person's feedback by physiological sensors [13], and in such cases, the data might not be protected at all.

Less sensitive data, but also of high interest and economic value to diverse organisations and different people, are collected for personalisation purposes, stored either on a personal device or in a central database (e.g., customers' data are often stored in a retailer's database) and often exchanged for providing personalised services. This information includes user profiles created from the collected data about shopping behaviour; travel preferences; user profiles created from web surfing, watching TV; and from e-learning exercises. Such data can reveal a lot about a person's psychology, lifestyle, finances, health and intelligence, especially when matched with information from other sources.

Professional skills (usually stored in a central database) may not be regarded as very sensitive data, but they could be of high interest to terrorists searching for a chemist or computer specialist. Such data is less sensitive than a person's identity data. However, they are of high interest to many more people and organisations because the data has have a commercial value and because one does not need to be a criminal in order to benefit from collecting such data. It is also worth noting that information flow is usually asymmetric between customers and service providers: customers transmit their (sensitive) personal information to the AmI shopping and commerce system while the system provides mainly unproblematic (mass) data including product and price information.

Probably the least sensitive information presented in the scenarios is information about the infrastructure of smart spaces, locations of objects and ways to control the smart space remotely. However, this information may be useful to criminals for robbery or acts of terrorism. For example, when people leave home, they take their personal device assistants with them, and these assistants carry a lot of information about homes and people and provide easy remote access to home. This leaves a lot of possibilities to a malicious hacker to initiate arson or gas leakage remotely.

To summarise, since the boundaries between different environments become blurred (people work and buy things from home and on the move, make doctor's appointments and check children from work) and since continuous monitoring (which includes storage of data) of a person's health and actions becomes common, all kinds of information about the person can be acquired anywhere. Probably the home, as the most private environment where people feel most secure, and a personal device assistant, which is always with a person, have the most data about people's identities, personalities, health and finances. This creates a high risk when a personal device is lost or stolen.

Almost all analysed scenarios postulate or assume benefits of ambient intelligence while only a minority refers explicitly to the threats associated with AmI at the same time. In almost all cases, it is mentioned between the lines that also threats exist, because it is always assumed that it is necessary that data are collected from the user, processed and matched with other information. A few types of threats are evident from the scenarios – either explicitly or implicitly.

In general, people tend to accept new technologies without worrying much about privacy if they get sufficient benefits from them (e.g., use of mobile phones and GPS in spite of the risk of location tracking). Nevertheless, it is fair to assume that risks to privacy will be inevitably increasing in the AmI world and, consequently, privacy protection should be built into AmI systems rather than relying only on user control over personal data. While one should assume a certain awareness of users about information flows and control over those flows is needed, there is, at the same time, a belief that control should not impose an unacceptable burden on the individual [23].

To complicate the situation even more, privacy and ethics are person-dependent, culture-dependent and situation-dependent. Thus, the big challenge in a future world of ambient intelligence will be not to harm this diversity, which is at the core of an open society.

3.5 Personal Control over AmI Technology

The level of control that a person has over the AmI system may vary considerably depending of the application. AmI has a high control level when it acts on behalf of a person, e.g., it decides to reject a phone call or to forego transmission of personal information. AmI has a medium control level when it gives advice, e.g., to reduce car speed due to a road bend ahead. AmI has low control level when it only executes a person's commands.

In most scenarios of modern life and in all scenarios of the distant future, AmI has a high control level over security (in the form of access control to online courses, houses, cars, work, health data, payments, in passports and immigration control) and privacy issues (scenarios do not present explicit user interactions with AmI systems where the user is granted access rights and control over personal data, thus, it is assumed that AmI has high level control over privacy issues).

Applications where a person's life depends on AmI and where AmI has a high level of control include safe mobility, especially driving (AmI detects obstacles, controls car speed and ensures that the car stays on the road), health monitoring and detection of a health crisis (such as heart attack). Generally, in driving scenarios, it is not clear if users are free to organise their travel means, time and route. Scenarios of future health care raise a question about whether medical monitoring and diagnosis systems are transparent enough for a typical (often elderly) user to gain a full understanding about what kind of data are gathered, where they are stored and transmitted, and what happens with them.

An important feature of AmI with high-level control is personalisation, which can be applied for adjusting an environment (lighting, heating); for filtering of shopping advertisements and selection of TV programs. An important question about personalisation is, however, not the degree of AmI vs. personal control, but the question about who is in control of the AmI system. Whether in shopping, or in news filtering, or in recommendations about medicines, trips, etc., how are the user's interests protected and how is it ensured that information is objective? At the moment, privacy protection activists have severe doubts about the customer's control of AmI-enabled shopping services [24]. Since retailers are the owners and operators of AmI infrastructure and provide customer services, one could assume that they would like customers to have as little control over AmI as possible.

This might result in customers not wanting to use AmI-enabled services or products at all.[1]

The AmI control level is also high in communications, first of all, because AmI handles connections between numerous different networks and adjusts the contents to user devices. Second, many scenarios describe high control at the application level, e.g., in emergencies where the communication between the ill or injured person, the emergency centre and the various paramedics en-route is completely automated. Manual intervention is only allowed in a few cases and is limited to acknowledgements. The emergency scenario is thus dependent on a well-designed process chain and complete coverage of the country with an AmI infrastructure. However, it begs the question about whether the emergency system would continue to function properly if major components in the AmI network are destroyed (e.g., in a terrorist attack or by natural catastrophe). Otherwise, this would suggest that, at the least, robust and possibly redundant communication procedures are needed that can also rely on low technology.

A foretaste of the high level of control in the communications domain is provided in the recent EU-level decision to require telecom operators to customer data for up to 10 years. Interestingly, civil liberties groups who opposed the measure were joined by the telecom operators themselves, not because of the potential for violation of civil liberties, but on cost grounds. The operators opposed the measure because they would bear the brunt of the cost of storing such data, which according to some estimates could increase ten-fold.

In some scenarios, AmI controls phone communications; it makes decisions about whether to connect a user with the calling person or not. In the ISTAG 'Dimitrios" scenario [1], this AmI functionality is presented most clearly: the personal device assistant can even mimic his owner's speech and talk to callers on his behalf. In scenarios which describe "always on" video connection between two different locations, it is usually an AmI task to close the connection if predefined rules state that it is not desirable or needed anymore, or if it detects a privacy privacy-threatening situation.

In the work domain, AmI is broadly applied to working tools (simulators and documentation), and in this sense, it has a high-level control over the work because an individual's decisions are based on simulation results and automated recordings of meetings. Although AmI just presents simulation results, and the decision is left to a person, it raises a question about how well simulators can take into account complex real-world systems and predict different situations, and whether people will rely too much on simulators instead of using their own imagination and creativity.

[1] The consumer protest about Sony's installing spyware on its CDs unbeknownst to consumers until Sony was exposed provides an indicative example. When it was discovered, Sony claimed that the spyware was to prevent illegal copying of music. However, the protest was such that Sony said it would withdraw the offending CDs. However, some reports say that Sony was continuing (at least as at the end of 2005) to sell such CDs [25].

The issue framed by the question: "Who is in control?" must also take into account situations where AmI technology may be used for surveillance. Instances of where the government and industry have engaged in surreptitious surveillance have come to light, e.g., Intel's encoding a unique serial number into every Pentium III microprocessor and the hidden files used to identify the authors of Microsoft Word documents [26]. One can distinguish two types of surveillance. The first type of surveillance is targeted. Targeted surveillance includes those situations where security agencies target suspected criminals or terrorists as well as surveillance cameras in the streets or shops. The second type of surveillance is not targeted. The latter type employs technologies that can be used, if needed, to identify someone. The existence or at least application of such technologies is not transparent to consumer (which distinguishes them from the surveillance cameras used to monitor motorists exceeding speed limits or used to apprehend shoplifters). The Intel numbering and Microsoft files and most spyware fall into the second category of surveillance. Of course, the surveillance may also be legal or illegal. Speed cameras are legal. Spying on citizens without a warrant is usually illegal, but even government agencies may engage in illegal activity.[2] Based on experience and history, one can only assume that some AmI technologies will also be used in a non-transparent and/or illegal way for the nefarious purposes of government and industry even when such surveillance is not warranted. Thus, even in situations where consumers think they are in control, they may not be.

4 Conclusions

The main conclusion from our analysis is that ambient intelligence technology goes beyond most of currently existing privacy-protecting borders.

First, increased connectivity between people and spaces blurs physical borders of observability such as walls and doors. A well-known example for this development are experiments in computer-supported collaborative work, namely, installation of video cameras in offices with the goal to increase awareness between colleagues and make communications between them more natural and more frequent. These experiments have shown that people forget easily about always-on video cameras, especially because the intuitive expectation "If I can not see you, then you can not see me" does not apply to computer-mediated communications [28], and this threatens personal privacy.

Second, the physiological sensors, always on, always attached to a person (whether for the goal of health monitoring or for personalising TV and learning programs), make this person absolutely helpless to hide his/her feelings because feelings can be discovered from changes in physiological parameters [29]. This means that facial expressions do not constitute a natural border protecting true personal feelings anymore.

[2] Recently, the Associated Press reported that "The [US] National Security Agency's Internet site has been placing files on visitors' computers that can track their Web surfing activity despite strict federal rules banning most files of that type" [27].

Third, the blurring of boundaries between time and space, recording and storing many kinds of information in AmI systems and increased capacity of data mining algorithms (which enable the finding of relationships and connections between diverse and seemingly unrelated pieces of data) violate personal expectations about spatial and temporal privacy-protecting borders, as well as expectations concerning ephemerality and transience of events.

New technologies will inevitably change personal expectations concerning privacy generally. Nissenbaum [17] cites a U.S. court decision as an example of such changes. The court decided that the police did not violate personal private space when they discovered an illegal activity while flying an airplane over a person's home and yard, because one cannot expect reasonable privacy from surveillance planes since flights have become a common part of our lives. So, what kind of changes in privacy expectations will replace the current expectations when AmI technologies become a common part of our lives? Whatever they will be, changes in people's expectations of privacy will happen more slowly than technology capabilities grow, as experiments in computer-supported collaborative work have shown.

The bottom line from our analysis of existing AmI scenarios is that they have tended to paint a rather too sunny view of the future. As an antidote, the SWAMI consortium has constructed several so-called "dark" scenarios, as we call them, which highlight things that can go wrong in the deployment and application of AmI technology [30]. From our analysis of existing scenarios as well as of our own dark scenarios, we conclude that a range of safeguards are needed to better protect privacy and re-assert greater user control [31]. Identifying and elaborating needed safeguards are the focus of our research in 2006.

Acknowledgement

This work is supported by EU project SWAMI: Safeguards in a World of Ambient Intelligence (IST-2004-006507). Important contributions to this work came from Ioannis Maghiros and Sabine Delaitre (EC/DG JRC, IPTS), Petteri Alahuhta (VTT Electronics) and Ralf Lindner (Fraunhofer ISI). The views expressed are purely those of the writers and may not in any circumstances be regarded as stating an official position of the European Commission.

References

[1] IST Advisory Group, Ducatel, K., Bogdanovicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: Scenarios for ambient intelligence in 2010. Institute for Prospective Technological Studies (IPTS), Seville (2001). http://www.cordis.lu/ist/istag-reports.htm

[2] Punie, Y.: The future of ambient intelligence in Europe: The need for more everyday life. Communications and Strategies **57** (2005) 141–165

[3] Weiser, M.: Some computer science issues in ubiquitous computing. Communications of the ACM **36** (1993) 75–85

[4] Bohn, J., Coroama, V., Langheinrich, M., Mattern, F., Rohs, M.: Living in a world of smart everyday objects: Social, economic, and ethical implications. Journal of Human and Ecological Risk Assessment **10** (2004) 763–786
[5] Ackerman, M.S.: Privacy in pervasive environments: Next generation labelling protocols. Personal and Ubiquitous Computing **8** (2004) 430–439
[6] Åkesson, K.P., Humble, J., Crabtree, A., Bullock, A.: Usage and development scenarios for the tangible toolbox. ACCORD Deliverable D1.3, Swedish Institute of Computer Science (2001). http://www.sics.se/accord/plan/del/D1.3.pdf
[7] Aschmoneit, P., Höbig, M.: Context-aware collaborative environments for next generation business networks: Scenario document. COCONET Deliverable D 2.2, Telematica Institute (2002). http://coconet.telin.nl/
[8] Harrop, P.: Item level RFID: The business benefits of the "tag everything" scenario. IDTechEx Ltd., Cambridge (2005)
[9] ITEA: ITEA technology roadmap for software-intensive systems, 2nd edition. Information Technology for European Advancement (ITEA) Office Association (2004). http://www.itea-office.org
[10] López de Vallejo, I.L.: E-locus: A clustered view of European ICT for future workspaces. E-Locus Deliverable D5.5, Fundación TEKNIKER (2004). http://e-locus. fundaciontekniker.com/
[11] Masera, M., Bloomfeld, R.: A dependability roadmap for the Information Society in Europe. AMSD Delilverable D1.1, Rand Europe (2003). https://rami.jrc.it/roadmaps/amsd/
[12] Morganti, F., Riva, G.: Ambient intelligence for rehabilitation. In Riva, G., Vatalaro, F., Davide, F., Alcaiz, M., eds.: Ambient Intelligence: The Evolution of Technology, Communication and Cognition Towards the Future of Human-Computer Interaction. IOS Press, Amsterdam (2005) 281–292
[13] Palmas, G., Tsapatsoulis, N., Apolloni, B., Malchiodi, D., Delopoulos, A., Beverina, F.: Generic artefacts specification and acceptance criteria. Oresteia Deliverable D01, STMicroelectronics s.r.l. (2001). http://www.image.ece.ntua.gr/oresteia/
[14] Riva, G.: Ambient intelligence in health care. CyberPsychology and Behavior **6** (2003) 295–300
[15] Savidis, A., Lalis, S., Karypidis, A., Georgalis, Y., Pachoulakis, Y., Gutknecht, J., Egger, B., Kramer, P., Tafra, M., Majoe, D., Lieu, V., Hunt, N., Gredmaier, L., Roberts, D.: Report on key reference scenarios. 2WEAR Deliverable D1, Foundation for Research and Technology Hellas, Institute of Computer Science (2001). http://2wear.ics.forth.gr/
[16] Lessig, L.: Code and other laws of cyberspace. Basic Books, New York (2000)
[17] Nissenbaum, H.: Privacy as contextual integrity. Washington Law Review **79** (2004) 101–139
[18] Singer, I.J.: Privacy and human nature. Ends and Means **5** (2001) 1
[19] Michahelles, F., Matter, P., Schmidt, A., Schiele, B.: Applying wearable sensors to avalanche rescue: First experiences with a novel avalanche beacon. Computers & Graphics **27** (2003) 839–847
[20] Friedewald, M., Vildjiounaite, E., Wright, D., Maghiros, I., Verlinden, M., Alahuhta, P., Delaitre, S., Gutwirth, S., Schreurs, W., Punie, Y.: Safeguards in a world of ambient intelligence (SWAMI): The brave new world of ambient intelligence – A state-of-the-art review. Deliverable D1 (2005) http://swami.jrc.es
[21] Friedewald, M., Da Costa, O., Punie, Y., Alahuhta, P., Heinonen, S.: Perspectives of ambient intelligence in the home environment. Telematics and Informatics **22** (2005) 221–238

[22] Elbirt, A.J.: Who are you? How to protect against identity theft. IEEE Technology and Society Magazine **24** (2005) 5–8
[23] Winters, N.: Personal privacy and popular ubiquitous technology. In: Proceedings of Ubiconf 2004, April 19th, Gresham College, London. (2004)
[24] Albrecht, K.: Supermarket cards: The tip of the retail surveillance iceberg. Denver University Law Review **79** (2002) 534–539, 558–565
[25] Farrell, N.: Sony gives rootkits for christmas. The Inquirer (26 December 2005). http://www.theinquirer.net/?article=28548
[26] Borrus, A.: The privacy war of Richard Smith. Businessweek Online (14 February 2000). http://www.businessweek.com/2000/00_07/b3668067.htm?scriptFramed
[27] Associated Press: Spy agency removes illegal tracking files. The New York Times (29 December 2005). http://www.nytimes.com/2005/12/29/national/29cookies.html
[28] Bellotti, V., Sellen, A.: Design for privacy in ubiquitous computing environments. In: Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), Dordrecht, Kluwer (1993) 77–92
[29] Nasoz, F., Alvarez, K., Lisetti, C., Finkelstein, N.: Emotion recognition from physiological signals for user modelling of affect. In: Proceedings of the 3rd Workshop on Affective and Attitude User Modelling (Pittsburgh, PA, USA, June 2003). (2003)
[30] Punie, Y., Delaitre, S., Maghiros, I., Wright, D., Friedewald, M., Alahuhta, P., Gutwirth, S., de Hert, P., Lindner, R., Moscibroda, A., Schreurs, W., Verlinden, M., Vildjiounaite, E.: Safeguards in a world of ambient intelligence (SWAMI): Dark scenarios on ambient intelligence - Higlighting risks and vulnerabilities. SWAMI Deliverable 2 (2005). http://swami.jrc.es
[31] Wright, D.: The dark side of ambient intelligence. Info - The journal of policy, regulation and strategy for telecommunications **7** (2005) 6 33–51

Profiles and Context Awareness for Mobile Users – A Middleware Approach Supporting Personal Security

Gerald Eichler and Matthias O. Will

T-Systems Enterprise Services GmbH, TZ ENPS, Deutsche-Telekom-Allee 7, D-64295 Darmstadt, Germany
{gerald.eichler, matthias.will}@t-systems.com

Abstract. This paper addresses the need for individualized information anytime and anyplace in contrast to the problem of user's right to privacy. The creation and application of profiles is characterized. A extensible middleware approach is proposed to combine independent databases on localization and personal profiles in order to identify the right layer to protect privacy. This will enable telecommunication providers to offer common basic services for the development of a wide range of mobile 3rd party applications.

1 Introduction: Pervasive and Personal Communication Services

While in the past, communication and entertainment devices such as ordinary telephones, television sets or personal computers were used by several-persons living together in a single place, mobile equipment (laptop, mobile phone, music player) reflects a social paradigm change towards personal devices & service consumption. More recently, community services e.g., *orkut* [1] or *openBC* [2] support sharing of individual experiences, using new channels beyond peer-to-peer-communication.

In this introductory section, some of the trends that are immanent in today's society are mentioned, which have to be reflected in the services, a provider offers to their customers. Common components may be shifted to a generic middleware platform, whose personalization services may be used by 3rd party application providers. However, technical implementation of profile-based services brings up the issue of application security, which has to be guaranteed, in addition to legal considerations.

Having sketched the preconditions to be fulfilled, section 2 will describe in more detail what personal profiles are for, how they are handled and securely stored. Section 3 will build on the notion of profiles to elaborate the idea of context modelling and awareness. It sketches the basic platform along with three function blocks that have been developed to support personalized services, including the interaction with resources by means of information tags. Section 4 elaborates on security requirements on various levels and proposes a generic architecture. The paper ends with a summary, emphasising preliminary thesis and results and giving an outlook on future work in the context of pervasive application service support.

1.1 About Paradigm Change

In the following, today's social and technical trends are briefly mentioned and discussed.

Communication and Information Technology Go Mobile. With miniaturisation, powerful processors, larger and brighter displays, compact and extensible storage capacity and the availability of several wireless access technologies at reasonable prices computing and communication needs are implemented in advanced mobile devices.

Private Meets Business. The strict separation of private and business communication devices is broken. Mobile devices turn into universal but personal tools to access private and business services. Common web access technology allows a smooth change between home and office networks.

Life Runs Faster. As users are faced with the need to spend more time with transportation on the way to work or while travelling, short time slots may be spent with the consumption of infotainment and edutainment offers. A mixture of compact, locally related information, advertising and entertainment clips can be enjoyed at the airports, in railway stations and on public transportation, offering new potentials for business models.

Users Want to Be Involved. "Interaction" is a key term, pushed by information and content providers to make better use of customer feedback via bi-directional communication systems. Typical push actions are replaced by dedicated pull actions to give the user the feeling to have control about the content, they are using. The more users are addressed personally, the more they want to be involved, e.g. individualized characters in mobile gaming.

Contents Feed Individual Requests. While newsletters may inform users about current events, they lack of context awareness and individual information provisioning. On the other hand, users may be reluctant in sharing their personal interests to an information provider due to the fear of abuse and spamming. The convergence of broadcast and IP networks [3] already allows efficient multicast technologies, to support semi-individual news distribution.

1.2 About Security

The issue about security is that nobody really wants to invest in its own protection, but takes it for granted that providers do. As long as no harm or abuse is experienced, users feel secure, although they are mostly not aware of potential risks. Firewalls, virus detection programs and spam filters are nowadays bundled with delivery packages. However, not ever useful default configurations are provided. More than two third of the users are not aware of how to make use of the most basic security features.

There seems to be a trade-off between simplicity of configuration and system safety. Security issues usually alert users at financial transactions or legal actions, but more often after a detection of a security shortcoming than in a preventive manner.

Individualized preventive behaviour would require a fair amount of configuration, which most users are not willing to invest.

1.3 About Technology, Law and Money

With multiple alternatively interaction and communication channels (WLAN, GPRS, UMTS) the acceptance is directly related to ease of usage of the associated technology. In this context, on demand selection of the right communication channel, depending on the infra-structural precondition, without requiring user interaction is required. Upcoming standards like media independent handover (IEEE 802.21) [4] try to fill this gap.

As in some countries, where UMTS license selling auctions were held, the exclusive right for the license holders to offer seamless mobile handover services leads to some political problems.

The typical user does not care about networks. He wants to get his information, in a certain place, at the right time and under affordable conditions, which means a trade-off between communication costs and service parameters e.g., transmission speed or tapping security. Simple charging models are required, e.g. flat rate models. Unfortunately, these are restriction to a certain access technology from a given provider and not related to the service a user is interested in. Often, cheap hardware and a service subscription are combined as a common practice of mobile network operators in countries such as Germany. However, service bundles are often not accepted, because the user has the impression of losing control over service pricing and potential additional charges. While providers are interested in selling long-term contracts, users would stay flexible enough to end a contract for any reason at any time. Thus, service providers have to think of acceptable business models which meet all requirements.

2 Personal User Profiles

This chapter presents an approach of creating and managing user profiles, as part of the user's context. This generates a prerequisite to offer truly personalized applications, whose behaviour reflect user's interests. Within the research field of "Pervasive communication", Deutsche Telekom is currently investigating a promising approach to offer seamless, personalized applications on top of a unified core framework [5]. The so called *Contigo Toolbox* (fig. 4) is basically a personalization engine for third-party application providers.

2.1 Purpose of Profiles

In order to offer personalized services, which go beyond location-based offerings, there is a need to be able to specify user interests on one side and resource characteristics on the other. Provided, that both refer to the same vocabulary, one can automatically induce how well a given resource is matching to the needs of a user. [6] Provided a given matching algorithm, which takes two profiles for input, compares the profile nodes and, optionally, semantic relationships between nodes, it returns a standardised matching value to the user, together with an optional list of terms that were significant in returning a given match.

Additionally, such profiles can be enriched with more user-specific information, such as location and time. However, in this context, the focus is on user interests and resource characteristics. As opposed to static, high-level profiles, which enable the user to select between a given number of topics, matching his interests, the approach for creating profiles allows for a much more flexible granularity, given a sufficiently expressive vocabulary. Thus, the more refined a user profile is, the smaller and more precise the list of matching resources will be. On the other hand, the more precise a resource profile is, the better it expresses a resource's characteristics and thus will allow the user in making a good choice from available service offerings.

2.2 Creation and Content of Profiles

Profiles are derived from a common vocabulary set. More precisely, the domain to be characterized (entertainment, sports, shopping, etc.) is modelled as concepts augmented by (symmetric) relations expressing the relationship between topics. This results in the modelling of ontologies. Profiles, on the other hand are considered to be customized ontologies, i.e. each term and relation of a given ontology may be given a weight within a (symmetric) interval [-1, +1]. Thus, each user and resource registered with the system is assigned its own profile, derived from a common basic ontology.

As resources and users are comparable with regards to their profiles, the example of a profile describing the characteristics of a user will be taken. These are identical to the set of user interests. Thus, a user profile is a composition of interests (fig.1).

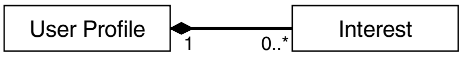


Fig. 1. A user profile as a composition of interests

A user can indicate his interest in a specific domain or even single concept by associating a value of a predefined range. Negative values indicate rejection of a given topic, while positive values show a user's interest in the associated topic. Whether the selection of values is discrete or continuous may depend on the domain and/or the requirements of a 3rd party application, and will not affect the platform's architecture.

As already mentioned, individual topics are interrelated, resulting in domain-specific relations, associated with a given topic set. These span a graph, which may be visualized as a tree with multiple inheritance, where the nodes represent topics (i.e. user interests or resource characteristics) and the edges represent semantical relations. Fig. 2 provides an example taken from the music domain to visualise the semantics of a set of given topics. In this example, titles are both being sung by the artist Madonna. One of the titles (Don't Cry for Me Argentina) belongs to two albums. In order to illustrate the weighing of profile items, consider a user who is interested in musicals (e.g. with a weight of 0.75), but does not particularly care for the pop artist Madonna (e.g. with a weight of -0.5). Thus, the expected result would be a weak recommendation of the Evita original soundtrack as the preference for musicals is more important in that case than the rejection for Madonna's music.

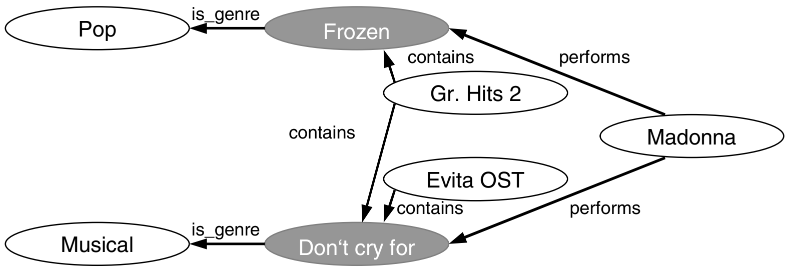


Fig. 2. Topics and relations shown in a example taken from the music domain

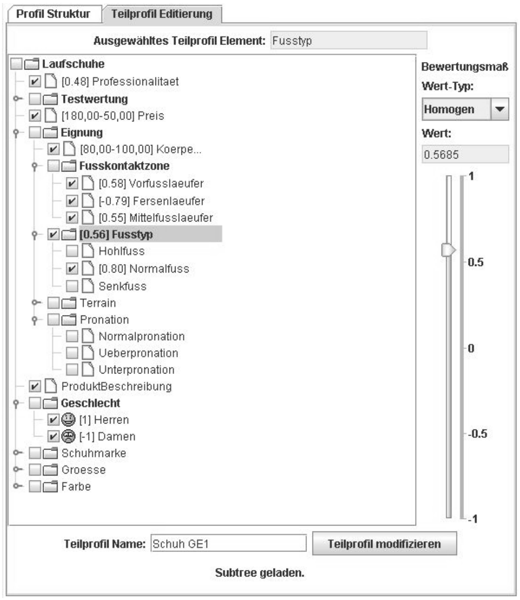
A title is associated with one or several music genres and has at least one artist interpreting it (uni-directed relations). An album is made up of a specific set of titles. As music portals also offer individual titles besides whole albums, a directed relation between a title and a potentially associated album is drawn. For instance, someone who likes a whole CD is likely to enjoy most of its titles, while a user indicating interest in a single title cannot be assumed to be in favour the album this title appears on.

While full ontologies as sketched above are very expressive, matching ontologies of this kind may be very resource-demanding, if you imagine a user in a shopping mall with potentially hundreds of service offerings, which would result in hundreds of matching processes (each resource against a given user).

Fig. 3 illustrates a user profile for the domain of running shoes. On the left side is a hierarchy of potentially interesting concepts for the user. Some of them are associated with a relative weight, others, such as the shoe size or prize, are given a numeric value. The slider on the right side may be used to associate a positive or negative value for a given term (strong rejection to strong acceptance).

A profile is rather complex, which raises the question of how profiles for resources and/or users may be handled more easily. Thus, a text mining module capable of extracting a profile when pointed to a textual resource e.g., a product description or a user's homepage, is used. A manufacturer with a description of his portfolio may then have the system "learn" the profiles of individual products, provided that the description contains terms that can be mapped to concepts of the ontology where profiles are derived from. Similarly, a user may enforce the system to analyse the text of some appropriate web page and produce an initial profile reflecting his own interests.

Another approach is to select a generic profile from a given set of stereotypes (e.g. for soccer fans) as a template and then adapt it to one's own needs with a few clicks. Profile learning mechanisms based on text mining may also be offered as a value added service for content providers who often are not capable to assign their resources with metadata that reflect the expectations and needs of end users. On the other hand, many freely available resources may be accessed on the internet to extract this very valuable additional information. Finally, a user may then simply point to the resources of interest for automatic profile generation.



```
Profil Struktur | Teilprofil Editierung

Ausgewähltes Teilprofil Element: Fusstyp

☐ Laufschuhe                                  Bewertungsmaß
    ☑ [0.48] Professionalitaet                Wert-Typ:
    ☐ Testwertung                             Homogen
    ☑ [180,00-50,00] Preis                    Wert:
    ☐ Eignung                                 0.5685
        ☑ [80,00-100,00] Koerpe...
        ☐ Fusskontaktzone                     1
            ☑ [0.58] Vorfusslaeufer
            ☑ [-0.79] Fersenlaeufer
            ☑ [0.55] Mittelfusslaeufer
        ☑ [0.56] Fusstyp
            ☐ Hohlfuss                        0.5
            ☑ [0.80] Normalfuss
            ☐ Senkfuss
        ☐ Terrain
        ☐ Pronation
            ☐ Normalpronation                 0
            ☐ Ueberpronation
            ☐ Unterpronation
    ☑ ProduktBeschreibung
    ☐ Geschlecht
        ☑ [1] Herren                          -0.5
        ☑ [-1] Damen
    ☐ Schuhmarke
    ☐ Groesse
    ☐ Farbe
                                              -1

Teilprofil Name: Schuh GE1    Teilprofil modifizieren

Subtree geladen.
```

Fig. 3. User Profile represented by an applet tree for the domain of running shoes

2.3 Secure Storage and Handling of Profiles

Personal user profiles, as described in the previous sections, need secure handling to prevent unauthorized parties to access or modify this information. Only a trusted provider may offer appropriate security and privacy mechanisms that are required for this context. Therefore all profiles are stored in a central certified database. This represents a central point of attack to potential intruders, thus requiring suitable mechanisms to prevent security attacks. This risk, however, may be reduced to distribute the profile data on several servers with replication mechanisms for redundant data storage, as used in many middleware architectures and application infrastructures, e.g. in the *ARIADNE foundation for Knowledge Sharing* [7], provided that storage and transmission of any data between individual networked servers is secure and complies to the high privacy standards a customer would expect from his provider. Manual modifications of profiles are done by a web application using encapsulation based on certified applets (fig.3). Application developers are provided with a toolbox, providing a limited set of APIs, not touching core components directly.

Another reason for a server-based profile managing approach is that the computing power required for fast and efficient profile matching is not available on mobile devices and smartphones and a user can own multiple devices. The only communication between mobile clients and the middleware platform, the Contigo Toolbox, is the activation of a user's profile and the result from profile matching, which is a real number between 0 % (no similarity) and 100 % (best match).

In order to prevent unauthorized access, a further security level may be introduced by equipping the user with a RFID-based smart-card which serves to activate a given user's profile or to enable a function, which allows controlled levels of visualisation.

This would require a NFC enabled mobile device, which allows a user to pair his Contigo card with his Contigo-enabled mobile phone to activate special security or privacy-relevant functions.

Assuming that both, end users as well as resources may at some point be associated with (user and/or resource) profiles. On the other hand, the introduction of text mining algorithms also offers another means of profile comparison, which does not require any profiles at all, i.e. text to text matching. Imagine that a number of resources e.g., telephones are available, and a user would like to know how similar two given devices may be. Then, upon comparing the textual description of these two resources, it is also possible to return a level of similarity to the user, although it may be more difficult to deduce the reasons for the computed degree of similarity.

Another use case would be a scientific author who, having found a highly relevant paper, would like to get some hints about other similar scientific papers. As it is not assumed that conference papers have associated profiles, text-to-text matching is to be preferred over profile-based approaches.

3 Context Awareness

As the notion of context is rather fuzzy, some clarification as to the types of information needed to appropriately describe a user's current situation is required. Given that, the building blocks of the Contigo framework are described, i.e. basic applications which enable personalized services. Last, the concept of information tags and its potentials when thinking about seamless new applications within the area of pervasive communication is discussed.

3.1 A Definition of Context

Context-aware applications are aware of the current situation of the user, which is particularly relevant for mobile settings, where the user context may continuously change. This includes position given by geographical coordinates, as used in location based services (LBS), time, but also the user's preferences and interests. Of these, the goals are probably most difficult to determine without explicit user interaction. According to [8], context is "any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.". In [9] one find an attempt at defining several context categories, namely:

- computing context, i.e. the characteristics of the end user devices including communication channels, behaviour and costs,
- time context, i.e. current time, weekday, month, season,
- user context, i.e. preferences and goals as well as activities and state of mind,
- physical context, which is all about environmental parameters such as temperature, traffic density etc., and finally
- history context, which is basically a chronologically sorted list of other context variables having occurred in the past, such as user interactions.

This last category is particularly interesting, since it allows deducing user goals from past activities. Context-aware systems or applications are thus considered to be context-aware if they use "context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [8]. When talking about systems which offer personalization, this means they are capable of considering a user's interests, preferences and background knowledge, which is then reflected in a user-specific interaction behaviour [10]. On the other hand, if systems require mostly user interaction to customize their behaviour, they are considered to be configurable.

When thinking about the kind of context-relevant information may be taken into account firstly, there is a distinction between low-level and high level context information. Low-level context information can be classified as either primary context, namely time and space, which is used to isolate specific entities as relevant to the querying system or user, or secondary context, such as optical, acoustical, biological or environmental data. Secondary context introduces further information, which may be relevant or interesting to the user. Other types of secondary context are the speed, orientation or acceleration of a user. On the other hand, high-level contextual information may not directly be sensed but is derived by analysing several sources of primary context, such as time, a user's location and the entries in his calendar. Note that in this example, such an analysis is only reliable if the user's information as read from his calendar is reliable and accurate.

While primary context is rather easy to model (time and space, where the latter may be associated with environmental models, e.g. of a building which would then allow a system to determine the room the user is currently located in), several approaches to model higher-level context can be found, ranging from simple key-value association, SGML-based modelling languages (ConteXtML, CC//PP, UAProf), object oriented models, logic-based models, surface models, and ontology based models. This latter approach is used to model an appropriate view of the world with its semantical relations, which is then used by both customers and manufacturers to describe user preferences and resource characteristics.

Most application services and prototypes, which claim to be context-aware focus on location-based approaches. As an example, fixed or mobile tourist guides display relevant points of interest (PoI) to the end user while moving in an often unknown environment. In 1999, the *NEXUS project* [11] at the University of Stuttgart, Germany, started with the aim of providing a world model, including real-world and virtual entities as a foundation for mobile location-based application services. Some of the relevant research issues are modelling concepts and model management, integration of sensor networks, consistency mechanisms and communication and security issues.

3.2 Context Aware Basic Applications

The goal of the Contigo approach, when thinking of contextual application services is to allow the user to find interesting entities of the real world in his vicinity, which can be thought of as resources with potentially associated (customisable) information e.g., buildings, places, commercial goods. In the demonstrator which was developed, the user may annotate resources, i.e. to leave remarks, comments, ratings etc. with (physical) objects and places, which could be used in a later phase for profile modelling as described in chapter 2.2.

First of all, the **resource finder** assumes that a provider such as a retailer has a database, where each object is associated with a product profile describing it. In addition, each object in turn could be associated with an arbitrary large amount of background information. In that setting, the user would first of all be able to retrieve an ordered list of interesting resources, ranked by personal relevance and with the option to see, why a specific entry was proposed. In a second step, the information associated with the resources, which were presented to the user would be filtered against the user's preferences. The latter, of course, would only be necessary in case of a vast amount of associated information to a recommended resource.

Secondly, the **contact builder** assumes that each user of the system has an associated profile describing relevant issues that are weighed positively or negatively, as described above. Similar to the resource finder, it is not only possible to find persons, which match a given search profile (as opposed to one's own profile), but also to have the system find potentially interesting users by starting with a list of friends and finding out related persons filtered against one's own preferences.

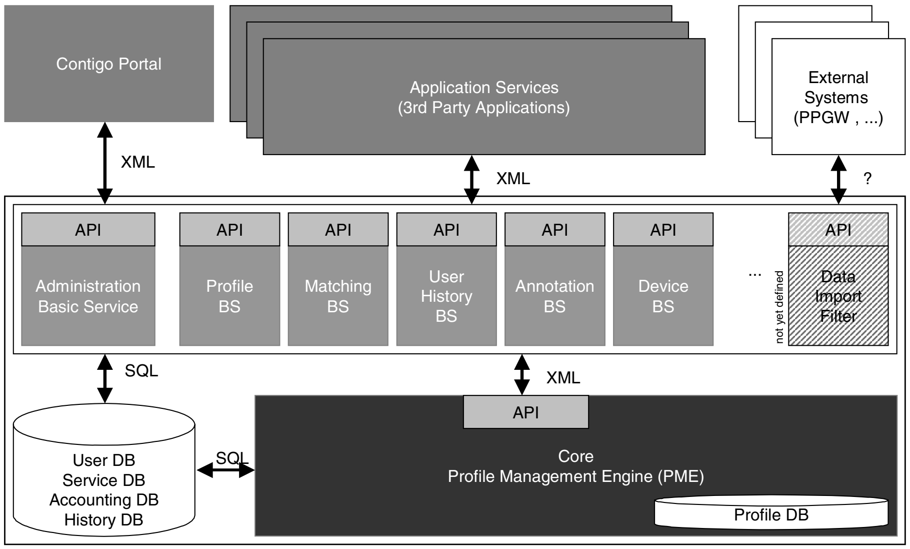


Fig. 4. Toolbox for contextual application services (Contigo)

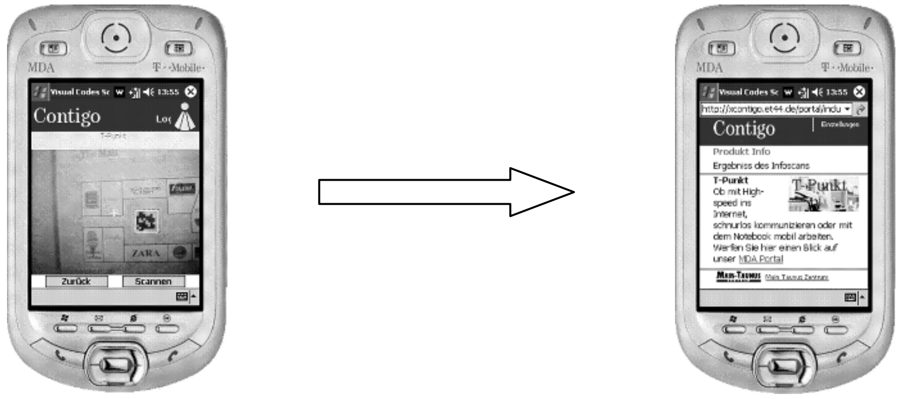
Last, the **annotation service** uses so-called information tags, i.e. points at which information may be exchanged between mobile devices and resources. For instance, a discussion forum about a specific location may be directly associated with a hot spot in the vicinity, or a new product may have an information tag where it is possible to leave a rating for the manufacturer. When considering innovative kinds of social software, such as weblogs or podcasts, this may very well alter the way of human-to-human communication.

As described in the previous section, an important part of user context, particularly when considering moving targets, is location. Thus, an architecture which allows unified localization from multiple sources is employed, i.e. the Permission and Privacy Gateway (PPGW), described in chapter 4.3.

3.3 Information Tags and Visual Codes

Pervasive communication takes users one step further from mobile communication and information exchange to interaction with real-world objects. This is made possible by introducing so-called information tags associated with specific points of interest and real-world objects. The simplest type are ordinary EAN barcodes, as found on books or most commercial products. A more sophisticated variation are visual codes, which are already used in online ticketing or postal services.

Additionally, they allow providers to associate a visual code with a map, divided in several active regions so that users may interact with such a map similar to image maps on the World Wide Web, by focusing a region with a built-in camera and then retrieving associated information, which may be filtered against one's own profile. By setting the camera under a certain angle, different actions e.g., buying an article, forward information to a friend, or getting more details that are associated with a given resource can be accessed.



T-Punkt
Ob mit High-speed ins Internet, schnurlos kommunizieren oder mit dem Notebook mobil arbeiten. Werfen Sie hier einen Blick auf unser MDA Portal

Fig. 5. Scanning a 2D visual code from a shopping mall's poster for personalized offerings. (a) scanning a visual map. (b) getting related information.

It might be possible to start discussions or mobile blogs or find other persons having accessed the same map or map region, and finally, with the annotation facility, one may also leave personal annotations for people sharing similar interests, also based on profile information. Finally, interaction via information tags also offers the possibility to download dynamic and interactive media by interacting with a poster or printed media, including payment processes, yielding a highly user-friendly interaction [13].

4 Security Requirements and Solutions

There is always a trade-off between the customer's desire for individualized and personalized information and the protection of privacy. User profiles are the crucial point within the proposed middleware based solution.

4.1 Trusted Partnership

When looking for personal information services, multiple players are to be considered, depending on the service architecture, namely the content owner, the broadcaster and the broadcast network operator, the application service provider (ASP), the Internet service provider (ISP), and the mobile network operator (MNO).

Finally, often a third party billing service provider is involved as a trusted party. This multi-party approach does not really encourage customer's trustfulness. If personal issues are concerned, users usually look for a single trustable partner. Customer acceptance tests have shown that the MNO has the closest relationship to mobile users, as it already owns the roaming information and is to be considered a reliable billing service partner. While there is a substantial range of applications, thinking of LBS the localization itself, implying track and trace is the most crucial point.

Secondly, interviews have shown that users require services to be transparent in behaviour and deliver exactly the functionality and information that is expected. This leads to an approach to composite context aware services which can be combined in a modular way, thus allowing to reflect customers' needs without unwanted add-ons. Users thus know what they are paying for and can rely on the provider only using exactly the part of personal data that is needed to fulfil the requested service.

Telecommunication operators, especially MNOs are in the position to have a close customer relationship to their end users. Mobile technologies are considered to be innovative and young people, requiring other applications than "just" telephony, are a new target for telecommunication services. Labels such as T-Mobile are already a trusted brand in Germany as well as in other European countries and the U.S.

Secondly, MNOs influence strongly the development of mobile devices and are already well experienced in the creation of bundles with hardware, services and medium and long term subscription contracts.

4.2 A Distributed Puzzle

The *DAIDALOS project* [12] proposes a separation of knowledge by restricting access to user information by means of using of multiple different identities along the following dimensions:

- changing identities over time,
- changing identities over services, and
- changing identities over providers.

Thus, unwanted tracing of user activities is complicated as required data is split into several independently stored segments without direct correlation. A combination of virtual ID's is a further step to increase secure access.

On the other hand MNOs try to complicate passive LBS. They will use virtual cell IDs in the near future to be independent from third party databases. This trend is well known from IP addressing issues.

Distributed data storage and its recombination with several data base knowledge on demand, decoupling the information tag and related information is an advantage from a privacy point of view.

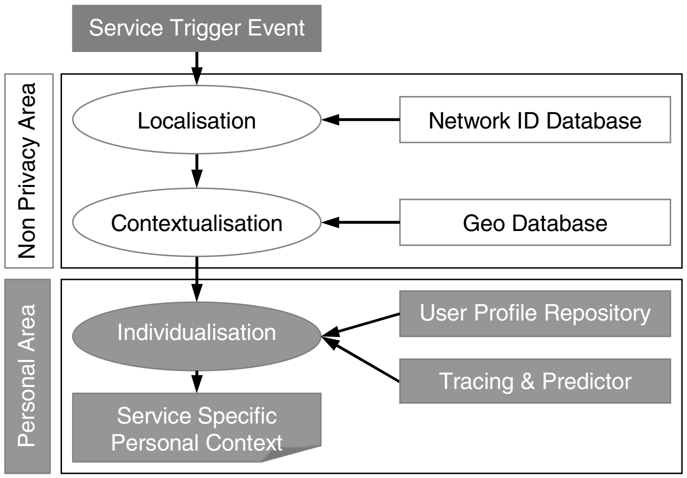


Fig. 6. Step by step context awareness with decoupling of information sources

Unified location sources, providing geo-coordinates irrespective of the detection technology allow for decoupling from an individual LBS provider [14]. Passive technologies like GPS or tag detection with near field communication access (NFC) omit central tracking. Network access detection (WLAN) or cell ID recognition (GSM) requires additional ID databases for geo-coordinate conversion. By combining multiple localization approaches, it may also be possible to achieve more precise localization, which is particularly interesting for rural areas, where single-source location is rather inexact, or even in in-house scenarios.

An independent hosting of a geo database (points of interest) transfers pure localization into context knowledge. In combination with profiles context awareness for a certain user is created. Fig. 6 illustrates this process.

4.3 The Mediator Approach

All the above issues lead to a middleware approach, operated by a trusted party, as user identity protection is highly important. Service usage metering is only collected for billing purposes, which is performed independently from service creation and operation. The DAIDALOS approach [12] mentions these functions as MARQS:

– Mobility management,
– Authentication, Authorization, Accounting (AAA),
– Resource management,
– Quality of service (QoS), and
– Security.

The framework, as shown in Fig. 7, bundles all general basic functions from the Contigo approach to run a business in the building black called “Common Basic

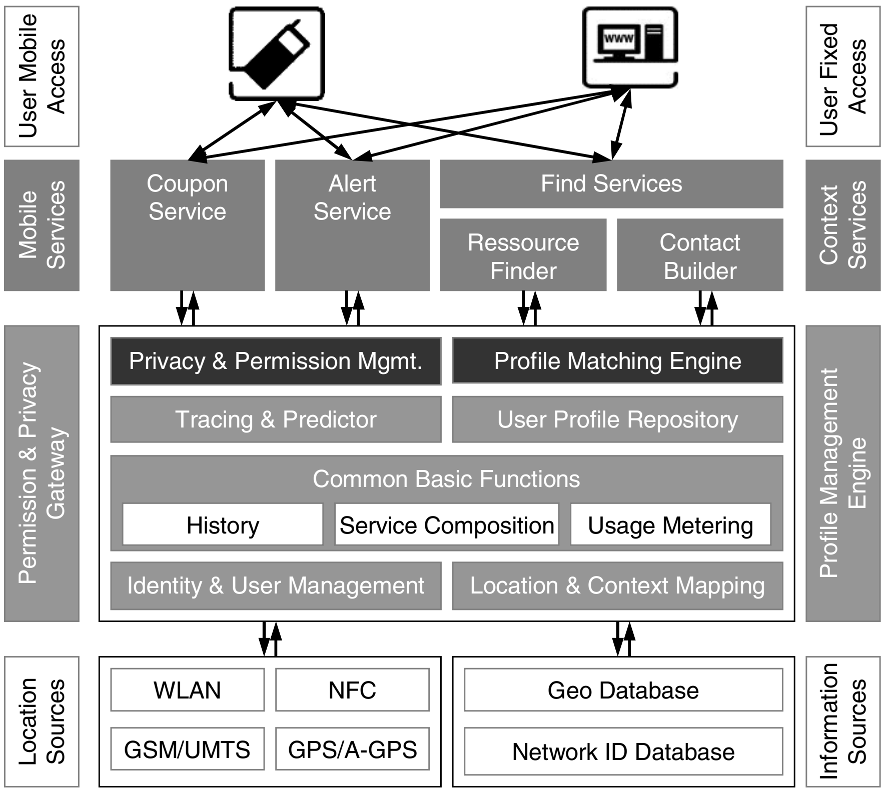


Fig. 7. A framework for middleware driven approach to decouple pervasive functions from user related private data

Functions", which indicates only some examples. Towards application level, which implies both, mobile and context services, two parts with privacy-relevant functions are found:

– the "Permission and Privacy Gateway" function (PPGW) and
– the "Profile Matching Engine" function (PME).

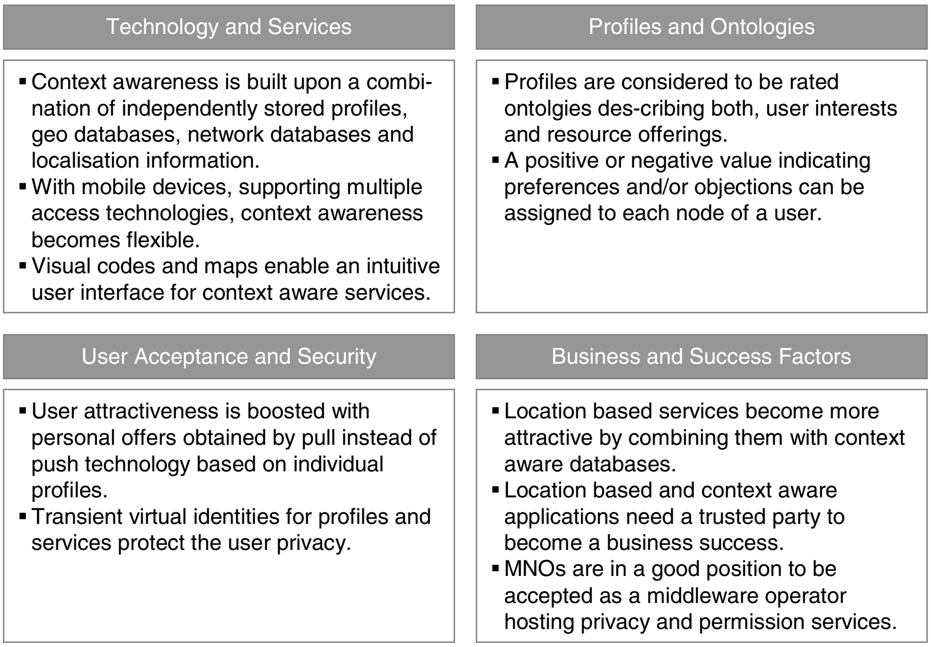
The PPGW decouples service specific requests, such as localization, from the user data. Based on virtual IDs tracing functions, including predictions, can be delivered to the application level. The PME provides a general filtering against the user defined interests. As described in chapter 2 profiles are matched against each other to detect a high grade of similarity. Thus the PME does not have to be aware of the association between profiles and resources (or users). The profiles are stored within the middleware at a "User Profile Repository" for two reasons, firstly to hide them against third party applications on a trusted level, and secondly to support high performance of agent based matching functions.

5 Summary

This paper has shown that in merging multiple technological approaches a wide range of context aware applications is made possible. User acceptance including profiles is mainly driven by a feeling of protected personality. A middleware approach can act as a mediator to serve both.

5.1 Preliminary Results

The preliminary results experienced during the project phases up to now are clustered in four blocks of thesis (fig. 8).



Technology and Services

- Context awareness is built upon a combi-nation of independently stored profiles, geo databases, network databases and localisation information.
- With mobile devices, supporting multiple access technologies, context awareness becomes flexible.
- Visual codes and maps enable an intuitive user interface for context aware services.

Profiles and Ontologies

- Profiles are considered to be rated ontolgies des-cribing both, user interests and resource offerings.
- A positive or negative value indicating preferences and/or objections can be assigned to each node of a user.

User Acceptance and Security

- User attractiveness is boosted with personal offers obtained by pull instead of push technology based on individual profiles.
- Transient virtual identities for profiles and services protect the user privacy.

Business and Success Factors

- Location based services become more attractive by combining them with context aware databases.
- Location based and context aware applications need a trusted party to become a business success.
- MNOs are in a good position to be accepted as a middleware operator hosting privacy and permission services.

Fig. 8. Clustered thesis for profiles and Context awareness mobile applications based on a middleware approach supporting personal security

5.2 Further Work

From a telecommunication's perspective, "pervasive communication" as well as "track and trace" are considered to be important innovation fields of the near and middle term research and business development. Therefore Deutsche Telekom Laboratories [5] support several innovation projects within these domains. Within all four divisions T-Mobile, T-Online, T-Systems, and T-Com value added services with pervasive components will be generated. The proposed common framework will be further developed step by step starting from the Contigo framework, enriched by security functions.

References

1. Online Personal Networking Service orkut. URL http://www.orkut.com/ and http://en.wikipedia.org/ wiki/Orkut/
2. openBC. URL http://ww.openbc.com and http://de.wikipedia.org/wiki/OpenBC/
3. Illgner, K.: What is needed for Mobile Broadcast Services. Presentation at 2nd DAIDALOS Public Workshop on Convergence of mobile and broadcast environments; Heidelberg, Germany, Oct. 19, 2005
4. IEEE Working Group 802.21: Media Independent Handoff Working Group. URL http://www.ieee802.org/21/
5. Deutsche Telekom Laboratories: Areas of Innovation and Research. URL http://www.deutsche-telekom-laboratories.de/english/
6. Jung, P., Lonthoff, J., Ortner, E., Will, M.: Matching User's Needs with Service Offerings To Enhance Customer Satisfaction and Social Networking. Submitted to UMUAI – The Journal of Personalization Research. Special Issue on User Modelling to Support Groups, Communities, and Collaboration
7. ARIADNE Foundation for the European Knowledge Pool. URL http://www.ariadne-eu.org/
8. Dey, A. K., Abowd, G. D.: The Context Toolkit: Aiding the Development of Context-Aware Applications. In: Human Factors in Computing Systems: HCI 99, pp. 434-441, Pittsburgh, 1999
9. Chen, G., Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College, Department of Computer Science, 2000
10. Fink, J., Kobsa, A.: User Modelling for Personalized City Tours. Artificial Intelligence Review, 18(1) p. 33-74, 2002
11. Universität Stuttgart, SFB 626: Nexus. Environmental models for mobile context-related systems. URL http://www.nexus.uni-stuttgart.de/
12. The DAIDALOS Project. Designing Advanced network Interfaces for the Delivery and Administration of Location independent, optimised personal Services. URL http://www.ist-daidalos.org/
13. Rohs, M.; Zweifel, Ph.: A Conceptual Framework for Camera Phone-Based Interaction. PERVASIVE 2005, Lecture Notes in Computer Science (LNCS) No. 3468, Springer-Verlag, Munich, Germany, 2005
14. Böhm, A., Leiber, T., Reufenheuser, B.: Trust and Transparency in Location-Based Services - Making Users lose their Fear of Big Brother. Conference presentation at Mobile HCI 2004, Glasgow, UK, Sept. 13 –16, 2004

Privacy Sensitive Location Information Systems in Smart Buildings

Jodie P. Boyer, Kaijun Tan, and Carl A. Gunter

Department of Computer Science,
University of Illinois at Urbana-Champaign, Urbana, IL 61801
{jpboyer, kaijunt, cgunter}@cs.uiuc.edu

Abstract. Increasing automation of buildings enables rich information streams about the activities of building users to reach networked computer systems. Privacy concerns typically cause this information to be accessible only by building managers and security personnel. However, if appropriate privacy mechanisms can be implemented, then it is possible to deploy location information systems that can contribute to the convenience and efficiency of users. This paper describes a three step approach to privacy-sensitive release of location information collected by building sensors. These steps entail defining an ownership model, defining environment events to be monitored, and creating a sharing model. These steps are described mathematically and then validated through a case study for a system called Janus's Map which provides a location information system for the card reader, door, and occupancy sensors of a modern smart building.

1 Introduction

Buildings are becoming increasingly intelligent through the deployment of Building Automation Systems (BASs). Advances in sensors, control systems, networking, and information technology enable greater automation to increase safety and comfort for users and convenience for building managers. BASs often also include enhancements to the security of buildings by providing more sophisticated entry mechanisms such as card reader systems and a variety of surveillance, audit, and remote control mechanisms. Such mechanisms create a rich stream of information about users. To address concerns about the privacy of users, it is typical to restrict this information to building managers and security officers, often with restrictions such as using surveillance data only to respond to specific types of security incidents. However, there are many cases in which the location information collected by the BAS can help building users as well as building managers. The aim of this paper is to present a model, architecture, and case study for distributing location data from a BAS infrastructure to users in a way that reasonably addresses privacy concerns.

For the purposes of this paper, let us define a *Location Information System (LIS)* to be a system that allows users and managers to control and learn information about tracked people and objects using information generated from a

BAS. Mechanisms that can provide the needed raw data include networked card reader systems, video surveillance cameras, and motion detectors. These may be supplemented by Radio Frequency ID (RFID) tags, computerized scheduling and workflow information, communication activity, and other mechanisms. As a running case study for this paper we have focused on a subset of such mechanisms that can be found in the Siebel Center in Urbana, Illinois, a "smart building" opened in 2004. The Siebel Center BAS audits information about the use of card readers to gain access to shared and individual offices throughout the building and supplements this with information collected from occupancy sensors and sensors that detect the state of doors. This information is used only by building managers to improve building maintenance functions (such as diagnosing faulty card readers) and respond to security incidents. It provides a good case study for the challenges of creating an LIS that would open this information to building users to support applications like team communications and improved workflow.

There are two key challenges that must be addressed to create an LIS from the existing infrastructure provided by a BAS. First, it is necessary to convert the information provided by the BAS into the kind of information needed for a useful LIS. BAS sensor data usually provides only an estimate of the desired information about the environment that users care about. For instance, if a user was identified by a card swipe opening a door to a room with an inactive occupancy sensor and the sensor has been active ever since, then it is credible to conclude that that user is in the room. However, it is also possible that the user in question opened the door for a colleague and it is this colleague who is in the room, not the one who used the card reader. If the LIS is to be of any use it must be able to tolerate this type of inaccuracy or use supplementary information to disambiguate the sensor data. The second challenge is to create a privacy system that meets the constraints for enterprise purpose and user privacy norms while still serving a worthwhile function that all of the stakeholders can agree on. A particular observation is that there is no "one size fits all" solution for this, since these requirements will change between major types of users. For instance, the needs of a commercial enterprise are likely to differ from those of a university or government building and there could be great variation within these sectors such as the difference between a government public building and a military facility.

Our approach to building an LIS for a BAS is premised on the idea of delivering a limited degree of discretionary control to users. The primary contribution of this paper is to describe and validate a breakdown of this strategy into three fundamental steps:

1. Define an ownership model for system events.
2. Determine the environment events of interest and how to deduce them.
3. Develop a model for privacy-sensitive information sharing for these events.

The process begins by determining a way to associate system events such as card swipes or motion detector readings with users in a way that users can be said to "own" the data because it is *about* them. Such events may be too low-level to be of direct interest so the next step is to determine environment events or state that users really care about, such as whether a user is in his office, and define ownership for these. These events must be inferred from system events with an acceptable threshold of error. The final step is to describe the rules that a user can employ to control the distribution of environment events that are owned by her.

We elaborate this approach in two stages, first by defining the general approach mathematically as a family of structures and functions that capture the key concepts at a high level and second by instantiating this model in a case study for the Siebel Center called "Janus's Map". It uses the electronic door lock system and occupancy sensors to implement an LIS that provides users information about the presence or whereabouts of other users in the building. Additionally, Janus's Map allows users to specify rules describing the information others are able to determine about them. These rules are fairly intricate but give users fine-grained control not only concerning who can see their data, but also how accurately their location is revealed to other users.

This paper is organized as follows. In Section 2 we discuss possible infrastructure that can be used for LISs. In Section 3 we present a general mathematical model for developing LISs. In Section 4 we present an instantiation of the general model for Janus's Map which is followed in Section 5 by a discussion of the architecture and implementation of Janus's Map. In Section 6 we discuss related work and then we conclude in Section 7.

2 Infrastructure and Applications for Location Information Systems

One of the key contributions of LISs is that they are able to determine high level information about an environment using existing infrastructure. There are many different pieces of infrastructure that could be used to aid in generating these environmental events, including ones outside of the typical BAS infrastructure. Here are a few examples:

Door Lock System. Many buildings now use electronic door locks on some or most of the doors. Building occupants can use cards to unlock the doors to which they have access. As a result, the door lock system is able to (approximately) determine which users have opened doors in the building.

Occupancy Sensors. Many BASs include occupancy sensors in many or all of the rooms in a building. These are typically used to control the lights in the rooms in order to save electricity, but may be linked into the BAS audit functions as well.

Network Jack Activity. Because many buildings assign specific network jacks to specific users, network activity can be used to gain (partial) information about user location.

Application Software. LISs could take advantage of certain types of application software, some of which provide their own activity data. In particular, most instant messenger programs provide status information about users.

Video Surveillance. Another piece of infrastructure that could be used for an LIS are the security cameras in common areas around the building. If a security camera is trained on a door, we would be able to keep a running count of users in a room by watching people enter and exit the room. If the security camera system is equipped with facial recognition software, the system would be able to locate people even in common areas, allowing the system to provide more accurate information. This source of data is considered particularly privacy-sensitive, however, and introduces significant processing challenges.

Wireless Network. There are a variety of suggestions for using wireless networks to allow a laptop to determine its location. This process could be included in an LIS. This information would be most useful for an LIS if the MAC address or IP address of a wireless device could be easily associated with a specific user.

Global Positioning System (GPS). Despite the fact that GPS does not work well indoors, it could still be used to aid an LIS designed for a use on a campus or to tell if an individual has entered a building.

RFID Tags. An LIS could work with either active badges, which transmit signals using battery power, or passive badges, like those used in inventory control. New technologies are making it possible to locate objects within rooms by reading these tags. Some advanced systems allow objects to be located in rooms to within a few centimeters.

Telephone. If a user's office phone is in use, this provides evidence that they are in their office. This information is not easily integrated with BAS data in most installations but the rise of voice over IP could change this.

Our formalism, which we describe in the next section, is intended to extend to all of these possible sources of data and others we did not list or have not considered. Our case study is based on a realistic subset of these that can be found in a specific building. The Siebel Center is a 225,000 square feet facility opened in April of 2004. The BAS in Siebel Center is the Continuum system from Andover Controls. It uses both the an electronic door lock system and occupancy sensors. These two systems can be combined to estimate the locations of individuals in the building. The door lock system in the Siebel Center generates five events of interest which are shown in Figure 1. The occupancy sensors in the building can be queried to determine whether a room is currently occupied. The building uses card readers for access to sensitive areas including offices of individuals.

There are a variety of applications that an LIS could serve and we will not attempt to provide a list of these here. We focus mainly on the basic function

`Valid_Access`	The user's swipe was accepted, and the door was opened
`ValidAccessNoEntry`	The user's swipe was accepted but the door was not opened
`InvalidAttempt`	The user's swipe was not accepted
`Door_Ajar`	The door was opened
`DoorAjarCleared`	The door was closed

Fig. 1. The events available in the door lock system

of providing users with general information about the physical location of other users with a level of accuracy that makes the information useful. However, there is much more that one could hope to achieve in an ambitious program along these lines. For instance, an integration of calendar data, telephone and computer usage, and BAS sensor data might provide a sophisticated LIS that estimates availability of a user as well as physical location.

3 A General Model for Location Information Systems

As mentioned above, we envision that there is a three step process in developing an LIS. We will follow these steps as we present a general model for LISs.

Ownership Model. The first step in developing an LIS is to define an ownership model. From the ownership model derives a mechanism to understand how privacy preference can be applied an enforced. In this section we define the ownership model for a general LIS. Every LIS will have a set of users, U, a set of locations, L, and a linear ordered space, T which represents time. Additionally, every LIS has a set, S of system events, which are generated by the BAS, or other systems. Additionally, we define three functions that act on a system event $s \in S$. The function $\mathit{user} : S \rightarrow U \cup \{\perp\}$ is used to determine the users associated with a system event s. The function $\mathit{loc} : S \rightarrow L$ determines the location in which s occurred. The function $\mathit{time} : S \rightarrow T$ determines the time at which s occurred.

In an LIS, we use ownership to determine who has control over certain entities in the system. In general, there are two system entities to which we assign ownership, locations and system events. In an LIS, we define two functions that determine the owner of an entity. We use the function $o : L \rightarrow 2^U$ to determine the set of owners of a location. This ownership function will likely be a static mapping to a set of users that, for instance, have access to a room in the building. The function $\rho : S \rightarrow 2^U$ is used to determine the set of owners of a system event. We usually take the owner of the system event s to be $\mathit{user}(s)$, or the users associated with the event. If $\mathit{user}(s)$ returns $\perp$, then the owner of s will probably be $o(\mathit{loc}(s))$, the owners of the location in which s occurred. For example, if a system event is reading an RFID tag, the owner of the event might be the user that owns the tagged object. In contrast, if the event was the temperature reading for a room, the owner of the event might be the owner of the room. It is important to note that ρ should not be a static mapping as events are generated on the fly.

Definition. An *ownership model*, $\mathcal{O}$ consists of:

- U, the set of users
- L, the set of locations
- S, the set of system events
- T, a set of values with a linear ordering, signifying time.

together with

– $\mathit{time} : S \rightarrow T$ which determines the time an event occurred
– $\mathit{user} : S \rightarrow U \cup \{\perp\}$ returns the user (if any) associated with a system event
– $\mathit{loc} : S \rightarrow L$ returns the location associated with a system event
– $o : L \rightarrow 2^U$ returns the owners of a location
– $\rho : S \rightarrow 2^U$ returns the owners of a system event □

Environment Events. The main purpose of an LIS is to aggregate system events into information that can be easily understood by users. For example, it may take several RFID readings to determine the accurate location of an object. The average user would prefer to see the aggregated information then a list of low level events. We call such aggregated information environment events. The next step in developing an LIS is to determine an appropriate set of environment events. An environment event may be determined from many low level events, such as location, or deduces from a single event which may be determined by a single event. We call this set of environment events E.

We then define a function $\mathit{induce} : 2^S \rightarrow 2^E$ which determines the set of environment events that may be deduced from a set of system events. We choose to implement *induce* as a function that applies a set of deduction rules to the given set of system events and returns a set of environment events. These deduction rules are of the form

$$\frac{s_1, ..., s_n \in \Gamma \text{ such that } C}{e_1, ..., e_m \in \Sigma}$$

where $\Gamma \subseteq 2^S$, C is a set of conditions on $s_1, ..., s_n$ and $\Sigma \subseteq 2^E$. This rule means that the existence of a set of system events satisfying a set of conditions implies the a set of environment events. Additionally, the hypothesis may also contain some conditions on the set of system events.

Information Sharing. The last and most important step of developing an LIS is to develop a method for information sharing. Currently, system events are heavily protected in order to respect user's privacy. In order to accommodate this, we define two families of functions $\mathit{filter} : U \times U \rightarrow (2^S \rightarrow 2^S)$ and $\mathit{mask} : U \times U \rightarrow (2^E \rightarrow 2^E)$. Each one of these functions returns a function that describes how elements should be altered or removed to project a user's privacy. We call $\mathit{filter}_u^v : 2^S \rightarrow 2^S$ u's filtering policy for v, which is applied when u is a target of v's search. We call $\mathit{mask} : 2^E \rightarrow 2^E$ u's masking policy for v, which is applied when u is a target of v's search. We apply two policies to each search because it gives he user not only the ability to control what system events are used to determine environment events, but also gives the user the ability to restrict what environmental events are returned. We sometimes refer to the tuple (*filter*, *mask*) as the privacy policy for an LIS. The privacy policy is the final component that needs to be defined for an LIS.

Definition. A *location information system*, $\mathcal{L}$, between an ownership model as defined above and a set, E, of environment events consists of the following 3 functions:

– *filter* : $U \times U \rightarrow (2^S \rightarrow 2^S)$ defines a user's filtering policy.
– *mask* : $U \times U \rightarrow (2^E \rightarrow 2^E)$ defines a user's masking policy.
– *induce* : $2^S \rightarrow 2^E$ is a function that maps a set of system events to a set of environmental events. □

Additionally, we define a family of functions *reveal* : $U \times U \rightarrow (2^S \rightarrow 2^E)$. We define the function $reveal_u^v : 2^S \rightarrow 2^E$ as the composition of *filter*, *mask*, and *induce* as follows:

$$\begin{tikzcd}
2^S \arrow[r, "filter_u^v"] \arrow[d, "reveal_u^v"'] & 2^S \arrow[d, "induce"] \\
2^E & 2^E \arrow[l, "mask_u^v"]
\end{tikzcd}$$

The function $reveal_u^v$ is called when v is performing an action for which u is the target.

4 An Example Instantiation: Janus's Map

As described in the previous section, there are three steps in defining an LIS. In this section, we present an insanitation of an LIS for the Siebel Center called Janus's Map. The Siebel Center is equipped with an electronic door lock system and occupancy sensors. The electronic door lock system allows doors to be opened with a person's university ID and the occupancy sensors are designed to shut lights off in rooms in order to save electricity. We propose using these systems to build an LIS.

Ownership. We begin with the ownership model of Janus, which requires us to define spaces of users, locations, system events, and times. Users can be modeled with user IDs and times as real numbers. Locations include such things as offices (represented as numbers) but the space required is more subtle than that. To accommodate our masking policy, we introduce the set $\mathcal{G} = \{floor, wing, room\}$ which defines the possible granularities of locations in the building. We also define the sets $L_{floor} \subset L$, and $L_{wing} \subset L$, and $L_{room} \subset L$, which (strictly) contain the floor locations, wing locations, and room locations, respectively. Therefore any location l in the building could be defined as a tuple of type $L_{floor} \times (L_{wing} \cup \{\perp\}) \times (L_{room} \cup \{\perp\})$. System events, S, are tuples of type $(U \cup \{\perp\}) \times L \times T \times \tau$ where τ is a set of types of system events comprising the following values: ValidAccess, ValidAccessNoEntry, InvalidAccess, DoorAjar, DoorAjarCleared, OccupancySensorTrue, and OccupancySensorFalse. These correspond to the event types in Figure 1.

To complete the definition of the ownership model we need to define functions *time*, *user*, *loc*, o, and ρ. The function *time* returns the times at which a system event occurred; such as the time at which a door is opened. The function *user* returns the user field of an event. The function *loc* returns the location in which an event occurred. Additionally, we define the function *type* : $S \rightarrow \tau$ which

Time	**Location**	**User**	**Type**
1/1/2006 07:45	SC3405	Alice	InvalidAccess
1/1/2006 10:00	SC4105	Alice	ValidAccessNoEntry
1/1/2006 10:01	SC4309	Alice	ValidAccess
1/1/2006 10:01	SC4309	$\perp$	DoorAjar
1/1/2006 10:03	SC4309	$\perp$	OccupancySensorTrue

Fig. 2. An example stream of system events in Janus's Map

returns the event type field of a system event. For the formal model we use a static policy describing the ownership o of locations.[1] Generally, any person who has a desk in a room is assigned ownership rights over a room. For public spaces, we assign ownership to a system administrator. Finally we define the ρ function. By default, we assign event ownership to the user referred to in the event, for example the user who swiped their card into the door. If the user field of an event is $\perp$, we assign the owner of the event to be the owner of the location at which an event occurred. A formal definition of ρ is in Figure 3.

Figure 2 shows an example stream of system events in the BAS for the Siebel Center. The owner of the event in first line is Alice and the owner of the event in the last line is the owner of the room SC4309.

Environment Events. Now that our ownership model has been defined, we can define our environment events for for Janus's Map. The main goal of Janus's Map is to present location information to users in the building. As a result we define an environment event that describes the location of an individual. We define E as set of tuples $U \times L \times T \times P$ where $P := \{\text{In}, \text{Near}\}$. The P field of an event describes whether we are certain that a user was actually in a location or only near it, for example, if a user has an invalid swipe to a door, we can only say they were near that location at the time of the swipe but they never actually entered it. In Janus's Map we define other environmental events, but for simplicity we will focus on user locations.

As described above, we choose to define *induce* as a function that applies a set of deduction rules to a set of system events and determines a set of environmental events that can be inferred from the given system events. One such deduction rule states that if a ValidAccess event occurs followed by a DoorAjar followed by a OccupancySensorTrue event all in the same location we can deduce that the users who performed the ValidAccess was in the room at the time of the OccupancySensorTrue event as well as that there were near the room at the time of the ValidAccess event. Given this induction rule, the events listed in Figure 2 induce two environment events: (Alice, SC4309, 1/1/2006 10:01, Near) and (Alice, SC4309, 1/1/2006 10:03, In). In the interest of space, we omit the other deduction rules for Janus's Map.

[1] In the implementation this needs to vary over time since users will periodically get new offices.

```
Function $\rho(s \in S) =$
if $user(s) = \perp$ then
  return $o(loc(s))$
end if
return $user(s)$


Let $T_u^v$ be a set of times
Let $L_u^v$ be a set of locations
Let $\tau_u^v$ be a set of event types
{# The above three variable are set by the users
when defining their filtering policy}
Function $filter_u^v(\Gamma \subseteq 2^S) :=$
Let $R := \{\}$
for all $\gamma \in \Gamma$ do
  if $time(\gamma) \in T_u^v$ & $loc(\gamma) \in L_u^v$ & $type(\gamma) \in \tau_u^v$
  then
    $R := R \cup \gamma$
  end if
end for
return $R$
```

```
Let $g_u^v$ be some element of $\mathcal{G}$
Function $mask_u^v(\Sigma \subseteq 2^E) =$
Let $R := \{\}$
for all $\sigma \in \Sigma$ do
  Let $(u, (l_f, l_w, l_r), t, \upsilon) = \sigma$
  if $g_u^v = floor$ then
    Add $(u, (l_f, \perp, \perp), t, \upsilon)$ to $R$
  end if
  if $g_u^v = wing$ then
    Add $(u, (l_f, l_w, \perp), t, \upsilon)$ to $R$
  end if
  if $g_u^v = room$ then
    Add $(u, (l_f, l_w, l_r), t, \upsilon)$ to $R$
  end if
end for
return $R$
```

Fig. 3. Algorithms for ρ, *filter*, and *mask*

Information Sharing. We now define the privacy policy for Janus's Map. Users define a privacy policy by specifying two functions that describe how system and environment events should be revealed in order to preserve a subject's privacy. We define both the *filter* and *mask* functions inductively by defining a general version of the functions $filter_u^v$ and $mask_u^v$[2]. These functions are shown in Figure 3 and described below. In order to define $filter_u^v$, the user, u first defines the sets T_u^v, L_u^v, and τ_u^v which specify requirements on system events that may be released to v. The set T_u^v specifies the set of times during which the system events must occur in order to be released to v, the set L_u^v specifies the locations in which a system event must occur in order to be released to v, and the set τ_u^v is the types of system events that can be released to v. The function then proceeds to filter out events that do not meet all of these requirements and returns only the events that occur during a time specified in T_u^v, in a location specified in L_u^v and have an event type specified in τ_u^v. In order to define a masking policy, $mask_u^v$, user u must first select a value for g_u^v, which is an element of the set $\mathcal{G}$. The value specifies the granularity at which locations can be returned to v, for example, Alice might specify that Bob is only allowed to know her location by floor. The $mask_u^v$ function the proceeds to mask a location based on the value of g specified by u. By default, $filter_u^v$ and $mask_u^v$ always return the empty set.

For example, Alice could define a filtering policy, $filter_{alice}^{bob}$, in which T_{alice}^{bob} is defined as the set of times defined by the phrase "Any day between 08:00 and 17:00", $L_{alice}^{bob} = Ł$, and $\tau_{alice}^{bob} = \{\text{ValidAccess}, \text{DoorAjar}, \text{OccupancySensorTrue}\}$. Similarly, Alice could define a masking policy, $mask_{alice}^{bob}$, such that $g_{alice}^{bob} = floor$.

[2] The filtering policy and masking policy are simplified here for readability. A full treatment of these policies is given in Section 5.

We now define the function *reveal*. The function $reveal_u^v$ is a composition of $filter_u^v$, *induce*, and $mask_u^v$. In Janus's Map, *reveal* returns a set of environment events. Because Janus's Map is mainly a search system, our reveal function roughly translates into a search function. The most recent environment event returned by $reveal_u^v$ would be the system's best guess of u's location as returned to v. If we were to apply $reveal_{alice}^{bob}$ to the set of system events listed in Figure 2, the following environment events would be revealed to us: (Alice, SC4, 1/1/2006 10:01, Near) and (Alice, SC4, 1/1/2006 10:03, In). We now present a discussion of the architecture and implementation of Janus's Map for the Siebel Center.

5 Janus's Map

This section focuses on the architecture of our prototype LIS that has been developed for the Siebel Center. Janus's Map can be broken down into 9 separate components. The interaction between these components is show in Figure 4. Breaking down Janus's Map into multiple components allows each component to be replaced if the infrastructure is changed with minimal effect on the other components.

User Interface. The user interface for Janus's Map is a web page. By using a web page, all users are able to use the system regardless of their preferred platform.

Location Service. The location service is the interface on the server side that services requests to the information service. The function *reveal* is defined in the location service

Data Cleaner. The data cleaner sanitizes the location data that will be returned to the users. It will be discussed in greater detail below. The data cleaner contains an implementation of *filter* and *mask*.

Access Control Module. The access control module is used to interface the the access control databases. The access control module contains an implementation of the function ρ.

Access Control Database. The access control database stores the users rules that grant the right of other users to see their data. These rules are discussed in greater detail below. Rules in the implementation of Janus's Map encapsulate both the filtering policies and the masking policies.

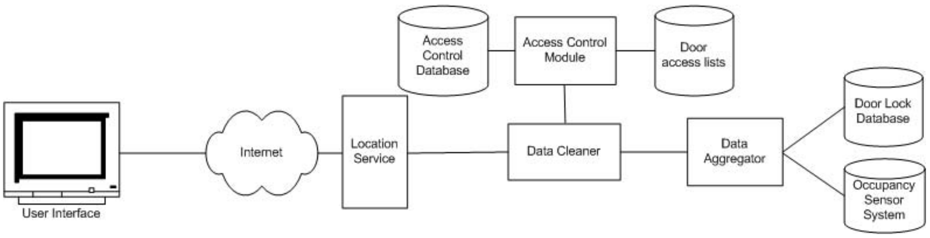


Fig. 4. The architecture of Janus's Map

Door Access List. The door access list stores the unlock rights for users for specific doors.

Data Aggregator. The data aggregator component collects data from the data sources and packages the data into a common data structure used throughout the rest of the system.

Door Lock Database. The door lock database contains a log of all events in the door lock system.

Occupancy Sensor System. The occupancy sensor system can be queried to determine the current occupancy state of a specific room.

Access Control. The access control system in Janus's Map is of key importance because it is a privacy sensitive system. As discussed in Section 4 users are able to define filtering policies and masking policies which control access to their owned events. In order to facilitate the delegation of access rights, Janus's Map allows users to define rules that encapsulate the definition of both the filtering policy and the masking policy. We give an example of a rule in Figure 5. Rules for Janus's Map can be split into 3 basic parts.

Targets. The targets of the rule are the users to whom the rule applies. For example, Alice has a rule with the targets Bob and Carol. This means that the rule will concern Bob and Carol's ability to access Alice's location data. This essentially allows users to define a single filtering and masking policy that applies to many users.

Data Access. The data access portion of a rule describes the kind of data that the targets can view about a specific user. Users are able to limit the quantity, the event type, the time, the date, and the room number. For example, Alice's rule states that Bob and Carol are only able to see events that occurred between 8am and 5pm on weekdays. This part of the rule defines a filtering policy.

Visibility. The visibility portion allows users to force certain fields to be hidden in the results returned to the rule targets. Additionally, the visibility portion of the rule also allows the users to change the granularity of the room data returned. Users can define the granularity to be one of 4 levels, room, wing, floor, or building. This way users are able to prevent the targets of rules

Target: Bob, Carol
Visible fields: Event Type, Event Time, Room
Granularity: Wing
Number of past entries: 5
Event types: Valid Access only
Event time: Between 8am and 5pm
Event date: From Jan. 1, 2006 to Jan 1, 2007
Event days: Monday - Friday
Rooms: All

Fig. 5. Alice's example rule

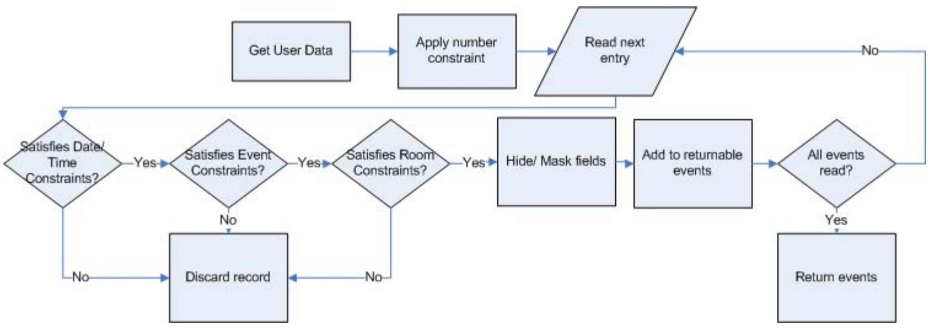


Fig. 6. The rule application process

from finding too much information about their location in the building. For example, Alice's rule states that Bob and Carol are only able to see the Event Type, Event Time, and the wing for events returned to them. This part of the rule describes a masking policy.

Rules Application. In order to fully understand how Janus's Map works, it is important to know the order each of the filtering rules are applied. The very first filter that is applied to the data is the number of past entries that can be used for deduction. For example, only the latest 5 events are allowed to continue forward. Once this happens, each event is then considered separately. The system determines if the date and time are within in the constraints, and whether the event type and event door are allowed by the rule that applies to the particular user. The last part of the rule that is applied is removing invisible fields and masking the room number as appropriate. This process is illustrated in Figure 6. You will notice that this process encapsulate both *filter* and *mask*.

The most important side effect of this ordering is that users are actually able to limit data sources. For example, if Alice says her friends cannot see when she is in the kitchen in wing 3B but also her friends are only allowed to know the wing of the room and not the actually room, her friends would never see if she was in 3B if the only room she ever used in the wing was the kitchen. One could imagine giving users the ability to limit types of data sources if many different types of data sources are used for the location detection, thereby giving the user direct control not only over the information other people see, but also the accuracy of the information.

Back Channels. One of the key challenges we encountered was preventing back channels. In information systems, users are often able to infer more information than they are explicitly given. An example of such a back channel involves the release of a rule. If Bob knew Alice had a rule that prevented him from knowing she was on the 4th floor, and he was returned no information when querying for Alice's location, Bob could guess with reasonable certainty that Alice was on the 4th floor. For this reason, users of Janus's Map are cautioned to keep their rules private. It may, however, still be possible for Bob to determine Alice's rules over time by aggregating personal observations with the information he was given by Janus's Map.

An additional back channel we face with Janus's Map is what we call "The Digital Rights Management Problem." One major problem with digital media is file sharing. As a result, many companies have made an effort to use technical means to prevent files from being shared, such as watermarking. However, once someone can execute the media there are many ways to circumvent this protection. One such attack would be hooking the speaker port into the microphone port and recording a new copy of the music that can be shared. Industry has, in effect, "raised the bar" on media sharing so that it requires more effort on the part of the media pirate. A location information system would be faced with a similar problem. For example, Alice states that Bob can only see her events on the day that they occurred. There is nothing to prevent Bob from using other means to create his own log of Alice's events over time. Therefore, in the system design it is important to "raise the bar" for this kind of attack.

We now discuss work that is related to Location Information Systems.

6 Related Work

We are not aware of any prior work that aims to provide a general formal system, methodology, and case study for exploiting a BAS to implement an LIS. The vast majority of work concerning BASs focus on administration as opposed to access by average building users. However, there is a substantial literature associated with each of the key components of this endeavor. We therefore survey some of the work on ownership, location detection, and privacy for location systems.

Ownership. The concept of ownership in the formalism here is technical not legal, but legal ownership does provide some guidelines for what the technical definitions might need to be, so it is worth making a very brief survey of some of these. In 1995, The European Union passed legislation concerning the use of personal information [1]. One of the most important parts of this law was that users had the right to object to the use of their information. In essence, the EU elected to give users ownership over their personal information, in that entities are required to ask users how to use their personal information. E-mail is assigned ownership in many jurisdictions including the U.S. Most service providers have a policy that states that they will not release a user's e-mails to anyone, giving the user ownership over their e-mails. If they would like this information to be read by other people, it is the user that is in charge of delegating rights to the e-mail, by forwarding the e-mail to others. Since the The Health Insurance Portability and Accountability Act (HIPPA) of 1996 was passed by the U.S. Congress passed, users have also been given some amount of control of their medical information such as the right to ask what sort of information is being stored about them and correct errors in this information. Additionally, the importance of ownership for location information has also been widely accepted. For example, the Wireless Privacy Projection Act of 2003 [2] requires cell phone providers to receive approval from users before their location information can be used.

Location Detection. Global Position System (GPS) [3] is one of the most common location detection systems. It uses satellites to determine location and works best outdoors when there is line of site to the satellites. Because of this, GPS is not effective indoors and cannot be used for location tracking buildings. AT&T Labs Cambridge developed the Active Badge system [4] and Bats [5] for indoor location detection systems. Active Badges transmit an identifier to building sensors using infrared and Bats perform a similar function but use ultrasonic pings. Both systems attach sensors to things that are to be located and also include infrastructure sensors. Active Badges provide room level accuracy and bats provide accuracy as good as four centimeters. These systems are in an experimental stage of development. In a separate project, we have been exploring the use of Ubisense (`www.ubisense.net`) as an LIS that handles both people and tagged objects and supports workflow monitoring. Nibble [6] is a location service that uses an 802.11 wireless infrastructure to determine room level locations. By measuring the signal strength from different access points, Nibble is able to estimate the location of the laptop on which it is running. This provides interesting potential as a foundation for an LIS, although it does require users to carry laptops and advertise the location information they collect.

Approaches for Location Privacy. A survey on this topic can be found in [7]. We augment this survey with some additional references. Although it does not concern location privacy *per se*, Graubart's Originator Controlled Access Control (OrCon) [8] is similar to the concepts used in Janus's Map. OrCon requires the data collector to request the originator's permission before data can be disseminated. OrCon seems to especially applicable to privacy systems because the originator and collector of data are often different. This is the case in the Janus's Map system where the originator can be consider the person who caused an event and the collector is the back end door lock system. Gunter *et al.* [9] present a formal privacy system inspired by and applied to location based services. This formal system allows users to issue licenses to allow subscribers to use the location information of the user. This approach is similar to the system presented here because both systems put the power in the users hands. The main difference between [9] and the current work is the level of detail at which the formalism describes how to create a license of this type. While [9] is quite general and aimed at many types of privacy-constrained applications, the model we have provided in this paper is focused on a specific application (LISs based on BASs) and provides somewhat more detail. Snekkens [10], takes similar approach to [9] to privacy policies; he recommends a central server stores information concerning user rules about the release of their location information and allows for fairly complex rules for users to specify the accuracy and detail included in released information. Like [9], Snekkens presents a fairly general system. Snekkens' focus is on the development of a language to define privacy policies while the current work focuses on mechanisms for enforcing privacy policies, in fact, one could imagine using the language presented by Snekkens to aid users in defining their privacy policies for an LIS.

7 Conclusion and Future Work

In this paper we investigate the use of Building Automation System sensors to build Location Information Services. Currently, BAS data is kept secret because of privacy concerns. For this reason, we consider privacy of paramount importance when developing an LIS. To this end, we presented a general mathematical model for developing privacy sensitive LISs. This model follows the following three basic steps: Define an ownership model, determine environmental events, and develop a system for privacy-sensitive information sharing. The most important focus of this model is to put users in control of data they own, because it is about them. The model allows users to define filtering and masking policies that control the access to their data. Additionally, because BAS events are generally low level, the LIS must also aggregate these events into environment events.

We then present an instantiation of this model for the prototype LIS, Janus's Map, to show the feasibility of creating an LIS for a building. Janus's Map uses the electronic door lock systems and the occupancy sensors to give users an approximate location of others in a building. As per the model, Janus's Map allows users to specify access permissions for events that they own. Users are not only able to specify who can find them, but are also able to limit the accuracy of any information released about them. In addition to discussing how this model could be instantiated for Janus's Map, we also present a discussion of the software architecture as well as a discussion of the challenges of preventing back channels in an LIS.

In the future, we hope to integrate more systems into Janus's Map. By doing this, we feel we could significantly improve the accuracy of system. Additionally, we would like to get feedback from users of the system as to its usefulness and accuracy. Additionally, because rooms have multiple owners, we would like to explore policy merging to resolve conflicting rules.

Acknowledgements

The authors would like to thank Adam Lee, Erin Chambers, Chuck Thompson and anonymous reviewers for their help and insightful comments. The work was sponsored by a grant from the MacArthur foundation and NSF grant CCR02-08996.

References

1. The European Parliment and the Council of the European Union: Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. (1995)

2. 108th Congress: HR 71: The wireless privacy protection act. In: United States House of Representatives. (2003-4)
3. Getting, I.A.: The global positioning system. IEEE Spectrum **30** (1993) 36–47
4. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. **10** (1992) 91–102
5. Ward, A., Jones, A., Hopper, A.: A new location technique for the active office. IEEE Personnel Communications **4** (1997) 42–47
6. Castro, P., Chiu, P., Kremenek, T., Muntz, R.R.: A probabilistic room location service for wireless networked environments. In: UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing, London, UK, Springer-Verlag (2001) 18–34
7. Görlach, A., Heinemann, A., Terpstra, W.W.: Survey on location privacy in pervasive computing. In Robinson, P., Vogt, H., Wagealla, W., eds.: Privacy, Security and Trust within the Context of Pervasive Computing. (2004)
8. Graubart, R.: On the need for a third form of access control. In: Proceedings of the 12th National Computing Security Conference. (1989) 296–303
9. Gunter, C.A., May, M.J., Stubblebine, S.: A formal privacy system and its application to location based services. In: Privacy Enhancing Technologies (PET). (2004)
10. Snekkenes, E.: Concepts for personal location privacy policies. In: ACM Conference on Electronic Commerce. (2001)

Silent Cascade: Enhancing Location Privacy Without Communication QoS Degradation

Leping Huang[1,2], Hiroshi Yamane[2], Kanta Matsuura[2], and Kaoru Sezaki[2]

[1] Nokia Research Center Japan,1-8-1,Shimomeguro, Meguro-ku, Tokyo, Japan
[2] University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo, Japan

Abstract. In a wireless communication network, it is possible to locate a user and track its trajectory based on its transmission, during communication with access points. This type of tracking leads to the breach of a user's location privacy. Prior solutions to this problem enhances user's location privacy at the expense of communication Quality of Service(QoS) degradation. In this paper, we propose *silent cascade* to enhance a user' location privacy by trading users' delay in silent cascade for anonymity. As a result, it avoids the problem of QoS degradation in prior solutions. Furthermore, we abstract silent cascade as a mix-network based formal model, and use this model to evaluate the performance of silent cascade. Study results prove the effectiveness of silent cascade under different types of QoS requirements. Besides, we also derive the optimal configuration of silent cascade to achieves target anonymity within minimum duration of time. and the theoretical upper bound of a silent cascade's anonymity.

1 Introduction

Recent technological advances in wireless location-tracking present unprecedented opportunities for monitoring the movements of individuals. While such technology can support many useful location-based services (LBSs), which tailor their functionality to a user's current location, privacy concerns might seriously hamper user acceptance. Some of those systems can accurately track users' position by overhearing other nodes' frames. We believe that those systems may cause serious location privacy problem on normal users of wireless communication.

In this paper, we concentrate on location privacy in the context of *trajectory privacy*. Trajectory privacy represents a user's right to prevent other parties from collecting a set of locations, visited by the user over a period of time. If an adversary is able to link two or more locations of an entity over time to build its trajectory, then it is able to profile the entity's behavior over time. This presents a threat to user's privacy. In our previous works [1, 2], we proposed a method *silent period* to achieve the unlinkability between users' two or more locations by forming a *mix-zone*[3] between nearby users. Analytical and simulation studies shows that this proposal enhances user's location privacy at the expense of communication gap.

However, typical communication applications (such as voice call) has strict requirements on the maximum gap in communication and/or minimum bandwidth. Hence, it may result in the degradation of the Quality of Service (QoS) if we allocates gaps in communication for location privacy. Therefore, current silent period can not satisfy some applications' contradictory requirements on communication QoS and location privacy. This restricts the application area of silent period. In this paper, we address the problem of *allowing any user to be able to achieve unlinkability between two or more of its locations in the presence of tracking by an adversary, without violating this user's Quality of Service requirements.*

Contributions of this paper are as follows. (1) We propose a scheme called silent cascade to enhance location privacy by trading off delay in silent cascade for anonymity. This proposal avoids QoS degradation in location privacy protection. (2) We abstract silent cascade into a mix-network based model. We use this model to derive optimal configuration of silent cascade, and upper bound of silent cascade's anonymity.

The rest of the paper is organized as follows. Section 2 describes the system model, the adversary model considered, and present QoS requirements of users' applications. Section 3 describes the proposed silent cascade protocol, models the silent cascade protocol into a mix-network based formal model, and presents the measures used in this model. Section 4 evaluates the performance of the proposed solution. Section 5 covers the related works, and Section 6 presents our conclusions.

2 System Model

Figure 1 illustrates a typical WLAN-based communication system. This system is conceptualized to comprise mobile station(MS), operator network, eavesdropper(E), and service providers on Internet. Operator network is composed of Access Point (AP) to provide wireless access service to MSs, gateway as the interface to service provider, access routers to forward IP packets between MS and service provider and several management servers. AP, MS, and E are incorporated with a WLAN radio interfaces operating at identical frequencies. A user carrying a MS moves on streets, and use wireless services provided by operator network to access service provider. MS updates its identifier periodically as a baseline scheme to protect its location privacy.

An E is capable of working in sniff mode, where it can capture all frames transmitted in the channel within its proximity. In addition, it is assumed that the wireless LAN interface in the E is capable of providing some radio metrics about the received frames to estimate the MS's current position. We also assume a global passive adversary(GPA) model. Such an adversary is able to overhear all the broadcasts of all the MSs, and hence able to estimate their locations. The purpose of adversary is to track the trajectory of the same node continuously when MS updates its identifier periodically.

Quality of service (QoS) is defined as the collective effect of service performance, which determines the degree of satisfaction of a service user. In a

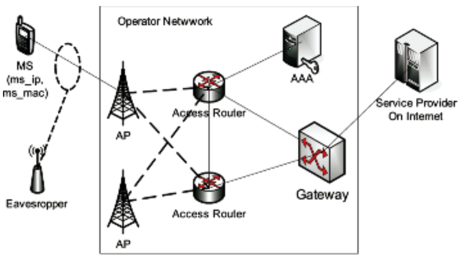


Fig. 1. Illustration of a WLAN Communication System

communication system, maximum gap within communication and bit rate of information are two key parameters impacting on user's QoS. Communication application can be largely classified into real-time application (e.g. VoIP, video streaming) and non-real-time application (Web browsing, Instant Messenger) based on their real-time requirement. Different kinds of applications has different requirements on those parameters. In principle, real-time applications are more sensitive to the change of maximum gap, while non real-time applications are more sensitive to the change of bit rate. For example, according to subjective assessment result of [4], maximum tolerable gap within a voice or video session is 200ms. Non real-time application such as ftp file transfer or HTTP web browsing can tolerate longer gaps, but but more sensitive to the loss of the effective bandwidth.

Our previous proposal enhances location privacy at the expense of losing communication time. Our studies show that silent period protocol can not satisfy some real-time applications'(e.g. voice call, video streaming) contradictory requirements on location privacy and QoS. This motivates us to develop a new scheme, which can provide location privacy without degrading application's QoS.

3 Silent Cascade for Location Privacy and QoS

3.1 Proposal: Silent Cascade

To satisfy users' contradictory requirements on communication QoS and location privacy simultaneously, we propose the new concept of a *silent cascade*. In the new proposal, a station has two operation modes: *active state* and *silent state*. The active state is defined as a state in which a station uses one specific pseudonym as its communication identifier; the silent state is defined as a state in which a station is not allowed to disclose either the old or the new pseudonym. *A silent cascade is defined as a duration of time that a station switches between silent state and active state periodically.* To guarantee applications' QoS, the maximum duration of time, by which a station can stay in either active or silent state continuously, follows those application's QoS requirement. The number of rounds of operation mode switch in silent cascade is a flexible design parameter.

Each round of operation mode switch creates unlinkability of a station's identifiers. By accumulating this unlinkability through multiple rounds of operation mode switch, a station enhance the the location privacy(unlinkability) between the last identifier before silent cascade and first identifier after silent cascade without violating application's QoS requirement.

We describe the detailed operation of a station using silent cascade as below. A station updates its pseudonym periodically to protect its location privacy. To increase the unlinkability between its old and new pseudonym, the station switches its operation mode from active state to silent state after each pseudonym update. After staying in silent state for certain period of time, the station switches back to active state for normal communication. Afterward, the station iteratively switches its operation mode between active state and silent state. Same as our previous silent period proposal, a silent state introduces ambiguity into the determinations of the time and/or place at which a change of anonymous address occurred. This creates a mix-zone between nearby stations. When a station is moving in 2-dimensional area, it meets different stations when entering silent state at different time instant. This creates a chain of *mix zone* with different station. To avoid the degradation of communication QoS, the configuration of a mix-zone–such as length of silent period and active period–follows application's QoS requirement. Although the maximum anonymity provided by a mix-zone may be trivial due to QoS requirement, the anonymity is accumulated when the station passes through a chain of mix-zone, and finally achieve target level of location privacy. Consequently, the silent cascade solves users' contradictory requirements of location privacy and communication Quality of Service (QoS).

We define *active period* T_a as a continuous period of time that a station stays in active state; define *silent period* T_s as a continuous period of time a station stays in silent state; define *lifetime* T_l as the duration of time of one round of mode switch, and it is equal to the sum of T_s and T_a; and define *length of silent cascade* as the number of rounds that a station switches its operation mode in silent cascade.

We illustrates an example of of silent cascade in right half of Fig. 2. A hexagonal station moves within a two-dimensional space. It enters silent state periodically, and meets stations with different shapes at different time instant. This creates a chain of *mix zone* between the hexagonal station and others. Anonymity is accumulated as the hexagonal station meets more stations in silent cascade.

3.2 Modeling Silent Cascade into a Mix-Network

When a station moves within 2-dimensional space and uses silent cascade protocol, it creates a chain of mix-zone. Chains of mix-zones created by a different users are interconnected to form a network. In this paper ,we abstract this network as a typical model used in anonymous communication: *mix-network* [5].

Fig.3 illustrates an overview of the abstracted system. A system $\mathcal{S}$ is abstracted into three entities (a set of mix-network users $\mathcal{U}$, a mix-network intruder $\mathcal{I}$, and a mix-network $\mathcal{M}$). The mix-network $\mathcal{M}$ consists of a set of mixes

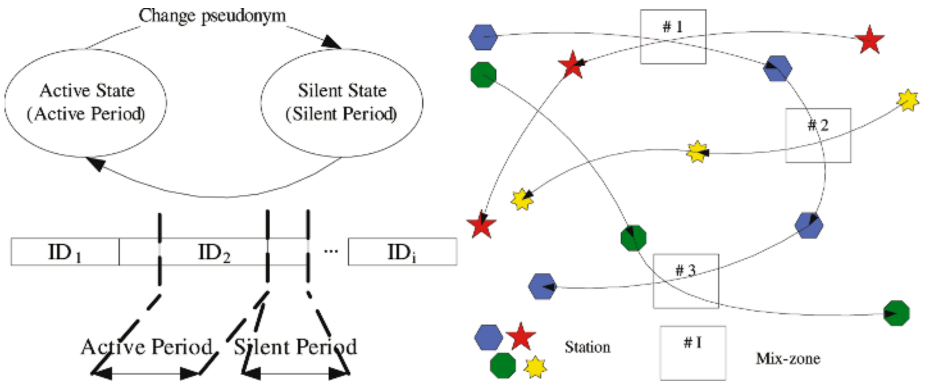


Fig. 2. Informal Description of Silent Cascade

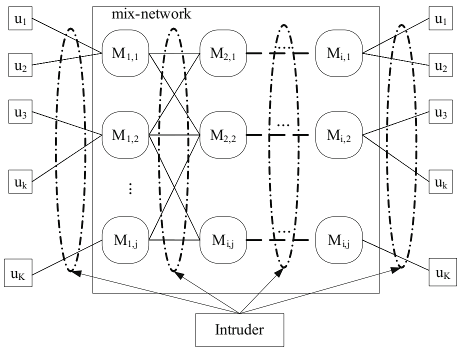


Fig. 3. MIX network formed by silent cascade

$\{m_{i,j}\} \in \mathcal{M}$. Each $m_{i,j}$ is formed by a subset of users. Function $u = USER(m_{i,j})$ returns the users of a mix $m_{i,j}$. The input and output of this mix are identifiers of users. Two subscripts of a mix $m_{i,j}$ are used to distinguish those mixes in the mix-network. We assume that all users enter mix-network at the same time, and use the same length of silent period and active period to form mix-zones. The first subscript i of $m_{i,j}$ represents the i-th round of mix in the silent cascade a user meets; the second subscript j is used to distinguish mixes in the same round. A $m_{i,j}$ includes a user's mobility pattern mm, traffic pattern tp, length of silent period T_s, length of active period T_a, and number of mix's input/output *density*. Furthermore, each user $u_k \in \mathcal{U}$ is capable of executing a set of actions $\mathcal{A}$ to the mix. From the intruder's point of view, these actions can be abstracted as a pseudonym $id \in \mathcal{ID}$. The output of one mix $m_{i,j}$ is the input of next round of mix $m_{i+1,j'}$. Regarding the capability of an intruder, we consider a global passive adversary in this system. It means that the intruder is capable of monitoring all traffics sent within mix-network and between users and the mix-network, but it can not inject any traffic into the mix-network or comprise any of the mix-zone. The functions within one mix-zone has been addressed in our previous publication [2], so we omit its details in this paper.

Here, we formally define the *silent cascade* of a user u_k as a subset of $\mathcal{M}$ that a user u_k passes through in the mix-network :

$$cascade(u_k) = \{m_{i,j} | m_{i,j} \in \mathcal{M} \wedge u_k \in USER(m_{i,j})\} \tag{1}$$

We define the length of a user u_k's silent cascade $length(u_k)$ as the number of mix in its silent cascade $cascade(u_k)$; the delay of a user u_k's silent cascade $delay(u_k)$ as the total duration of time a user spends on the silent cascade. There are two parts of delay in the mix-network: silent period, active period. We model silent period T_s as the batching delay inside a mix, and active period T_a as the propagation delay between two adjacent mixes in the mix-network. The topology of a mix-network depends on the mobility model of users. The effect of different topology on the performance of a silent cascade is out of the scope of this paper.

3.3 Measures of Anonymity, QoS and Delay

To perform a meaningful analysis, we need to have some measures for user's silent cascade. In our previous paper [2], we define a set of users which forms a mix-zone to contribute to target's anonymity as Geographical Anonymity Set(GAS), and use the entropy and size of GAS as measures for mix-zone. In this paper, we use the entropy and size of mix's GAS as anonymity measures. We define the size and entropy of a mix $m_{i,j}$'s GAS in the mix-network as $s(m_{i,j})$ and $h(m_{i,j})$.

We define the size and entropy of user u_k's silent cascade $cascade(u_k)$ as: $S(u_i)$ and $H(u_i)$ respectively. As described in [6, 7], the anonymity of a mix-cascade can be expressed by equations below if the inputs of any two mixes on the cascade are independent.

$$S(u_k) = \prod^{m_{i,j} \in cascade(u_k)} s(m_{i,j}) \tag{2}$$

$$H(u_k) = \sum^{m_{i,j} \in cascade(u_k)} h(m_{i,j}) \tag{3}$$

On the other hand, we use silent period T_s and *silent ratio* R_s as system's QoS measure. R_s is defined as the ratio of silent period T_s to lifetime T_l. It indicates the share of time a user spends on location privacy protection. Typically, silent ratio is used as a QoS measure for non-real-time application, while the length of silent period is used as a measure for real-time application.

The last issue is about the delay measures of the mix-network. As the length of silent cascade increases, anonymity and delay of a silent cascade increases simultaneously. Delay measures are designed to evaluate the duration of time to achieve a given target anonymity level. We use two anonymity targets in this paper: target entropy H_{target} and *Maximum Tracking Time(MTT)*. We define $TT(H_{target})$ as the duration of time from the beginning of silent cascade to the time when the entropy of silent cascade achieves H_{target}, and define MTT as the maximum duration of time from the beginning of silent cascade to the the first failure of tracking under a given tracking algorithm.

3.4 Comparison Between Silent Period and Silent Cascade

Comparing silent cascade with silent period protocol, we perceive mainly two differences. First, there are two trade-off parameters in silent period: anonymity and QoS; while there are three trade-off parameters in silent cascade: anonymity, QoS, and delay. Secondly, the objective of a silent period is to protect the unlinkability between users' identifiers, while the objective of a silent cascade is to protect the unlinkability between user's identifiers without diminishing QoS.

Left half of Fig.4 indicates the trade-off relation between anonymity and QoS in silent period. If the combination of an application(such as ftp file transfer)'s QoS and anonymity requirements are below the curve, it is possible to provide location privacy without diminishing QoS by silent period protocol. If the combination is located above that curve, it means that silent period protocol can not satisfy these two contradictory requirements.

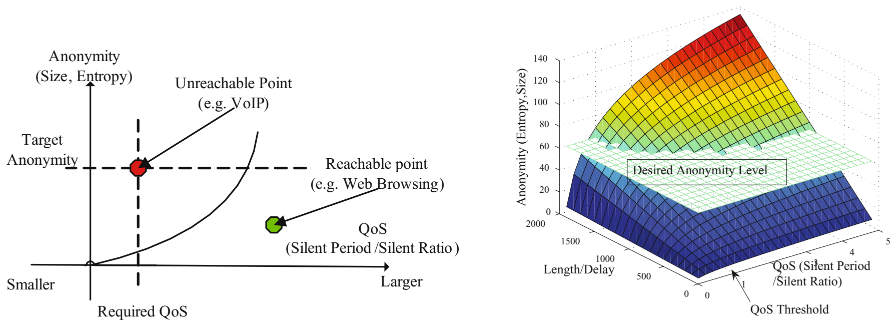


Fig. 4. (left) Illustration of trade-off relation between QoS and anonymity in silent period, (right) Illustration of trade-off relation between QoS, cascade Length, and anonymity

In silent cascade protocol, we solve these two contradictory requirements by introducing a new parameter: delay of silent cascade. Instead of trading location privacy with QoS, silent cascade protocol enhances users' location privacy by trading location privacy with delay of silent cascade. Right half of Fig.4 illustrates the relation between these three parameters. The slope in the figure shows the maximum anonymity can be provided under certain silent period(QoS) and delay condition. Horizontal plane shows an application's minimum anonymity requirement. By increasing the delay of silent cascade, anonymity provided by silent cascade can reach the horizontal plane without increasing QoS. A summary about the difference between silent period and silent cascade is given in Table 1.

Table 1. Comparison between silent period and silent cascade

Proposal	Features/Objectives	Solution(Mapped Formal Model)	Trade-off Parameters
Silent Period	Unlinkability between id	mix	anonymity, QoS
Silent Cascade	Unlinkability between id, communication QoS	mix-network	Anonymity, QoS, delay

4 Evaluation of Silent Cascade

In this section, we first introduce the simulation setup. After that, we analyze the performance of silent cascade when silent ratio and a silent period are used as QoS measures respectively. Finally, we discuss the upper bound of anonymity in silent cascade and its impact on location privacy protection.

4.1 Simulation Setup

In the simulation, a group of nodes move within a 2-dimensional area. They communicate with their access points with certain traffic pattern, and follow silent cascade protocol to periodically change MAC address and switch between silent and active states. Adversary randomly selects one node as target, and uses a tracking algorithm to track that node's movement.

First, regrading the traffic models used by stations, we take into account Voice over IP (VoIP) video streaming as the representatives of real-time application, and HTTP/web browsing, FTP file downloading as the representatives of non-real-time applications. A VoIP application generates fixed size packet(100bytes) with a constant interval (60ms). Besides,a video streaming application generate frames in constant interval (30fps) and fixed frame size (1600bytes). Furthermore, HTTP/Web browsing application is modeled as ON/OFF periods representing web-page downloads and the intermediate reading times. A FTP application keeps transmission of fixed size frame (1500bytes) continuously. The details about those traffic model follows the 3gpp recommendation [8].

Secondly, regarding the mobility of nodes, a set of nodes moves within a $180m \times 180m$ 2-dimensional area. *Random walk* are used as nodes' mobility model. In the random walk mobility model, a user first selects one direction between 0 and 2π, and a speed between 0 and 1 m/s, which are maintained for about 10 seconds. Afterward, the user iteratively selects its speed and direction. All stations use silent cascade when moving within simulation area. The effect of two parameters in silent cascade: silent period, lifetime, are evaluated in simulation. The behavior of mix-zones has been refined to take into account the effect of WLAN radio. In addition, silent cascade is used as privacy protection protocol. In addition to length of silent period and lifetime, we also assume that all nodes update their address independently. Furthermore, we assume density between $0.001/m^2$ and $0.1/m^2$, tracking accuracy between 0.1 and 2.0, lifetime between 25 and 200 are also evaluated in simulation.

Finally, we discuss the specification of the Intruder of a mix-network. The intruder is capable of monitoring all traffics within mix-network and between mix-network and users. We consider *simple tracking* as the tracking algorithm used in simulation. Simple tracking uses the length of nodes' silent period T_s, the maximum velocity of nodes v_{max} and the last observation of the user before entering mix-zone, to determine an area called reachable area where nodes may appear after silent period. Consequently. if there are n nodes in the reachable area, the probability that target's old and new identifier can be associated by adversary correctly is $1/n$.

4.2 Evaluation of Silent Cascade When Silent Period Is Used as QoS Measure

First, we analyze the performance of silent cascade protocol when the length of silent period is used as QoS measure. As discussed in Sec.2, typically the length of silent period is used as many real-time applications' QoS measure.

Left half of Fig.5 shows the growth rate of system's anonymity under different length of lifetime. We perceive that when the delay of silent cascade is small, the anonymity of silent cascade is trivial, but it accumulates in proportional with the delay of cascade(travel time).

Right half of Fig.5 illustrates the relationship between length of silent period and threshold time when the target size of silent cascade's GAS is 2.0, 5.0 and 10.0 respectively. From this figure, we perceive a clear trade-off relation between threshold time and length of silent period. It means it is possible to achieve the same target anonymity with tighter QoS requirement at the expense of delay in the cascade. However, from this figure, we also perceive that a station takes about 1500 seconds (25 minutes) to achieve target size of GAS 2 (tracking failure probability: 50%) when maximum silent period is 500ms. Comparing threshold time with the length of a typical phone call(3 minutes). it is obvious that most of phone call ends before silent cascade reaches a target anonymity level. Consequently, We think that our silent cascade proposal is effective for long duration

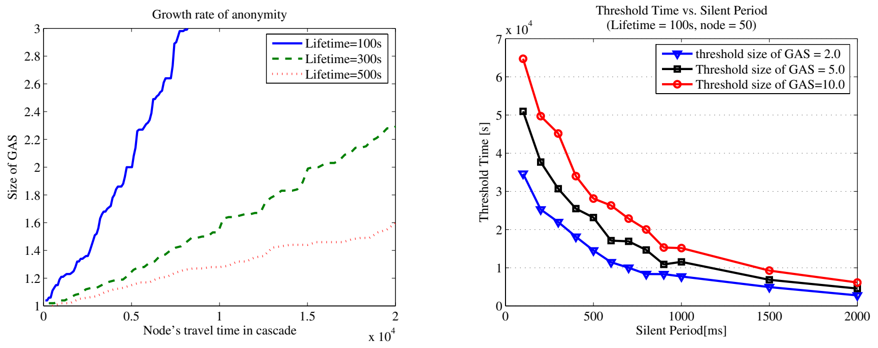


Fig. 5. (left) Growth rate of anonymity vs. size of silent cascade's GAS, (right) silent period vs. threshold time

real-time applications (e.g. 2 hour video streaming), but is not suitable for short period real-time application such as VoIP.

4.3 Evaluation of Silent Cascade When Silent Ratio Is Used as QoS Measure

In this subsection, we analyze the performance of silent cascade when silent ratio is used as QoS measure. First, we observed similar trade-off relation between silent ratio and anonymity as what we observed between silent period and anonymity. We omit those results in this paper because of page limit.

Secondly, we perceive that address lifetime has two contradictory roles on the delay of silent cascade. First, the anonymity of a mix-zone increases as the length of silent period increases. Because we assume that the ratio of silent period to lifetime is constant, longer lifetime results in larger anonymity of a mix-zone indirectly. However, lifetime also represents the delay between two round of mix-zone in the mix-network. If the length of a mix-network is constant, delay of a silent cascade increases as the increase of lifetime. Given lifetime's contradictory roles on system's anonymity, there may be optimal configuration by which silent cascade achieve anonymity target within minimum period of time. Below we first analytically evaluate the configuration of silent cascade under certain QoS and anonymity requirement, and then evaluate the results by simulation.

We define Maximum Tracking Round(MTR) as the number of rounds for which adversary can track a user continuously. Time from the start of tracking to the beginning of first mix-zone is considered as part of the tracking time. If adversary starts tracking randomly, expectation of the time nodes is tracked before first mix-zone is $T_l/2$. Consequently, expectation of MTT can be expressed as equation below.

$$E[MTT] = E[MTR] \times T_l + 1/2 \times T_l \tag{4}$$

In addition, we assume all mix-zones in the silent cascade has same size of GAS s_0, simple tracking is used as tracking algorithm. We define p_0 as the probability that adversary can correlate inputs and outputs of a mix-zone correctly. According to the last paragraph of Sec.4.1, p_0 is equal to $p_0 = 1/s_0 = 1/(D \times \pi \times (T_s \times v)^2$, given node density D, maximum speed of node v and length of silent period T_s. The probability that adversary succeeds in correlating first $(n-1)$ stages, but fails at $n-th$ stages is: $p_0^{(n-1)} \times (1-p_0)$. Consequently, given silent ratio R_s and lifetime T_l, the expectation of MTR and MTT can be derived as below:

$$E[MTR] \ = \textstyle\sum_{n=1}^{\infty}(n-1) \times p^{(n-1)} \times (1-p) \cong \frac{p}{1-p} \tag{5}$$

$$E[MTT] = \frac{a}{T_l} + \frac{T_l}{2}, \text{where } a = 1/(D \times \pi \times (c \times v)^2) \tag{6}$$

From Eq.(6) above, we can easily derive that given a combination of node density D, silent ratio R_s and speed v, MTT achieves its minimum value $\sqrt{2/\pi D R_s^2 v^2}$ when lifetime is equal to $\sqrt{2/\pi D R_s^2 v^2}$ and size of a mix-zone' GAS s_0 is 3.

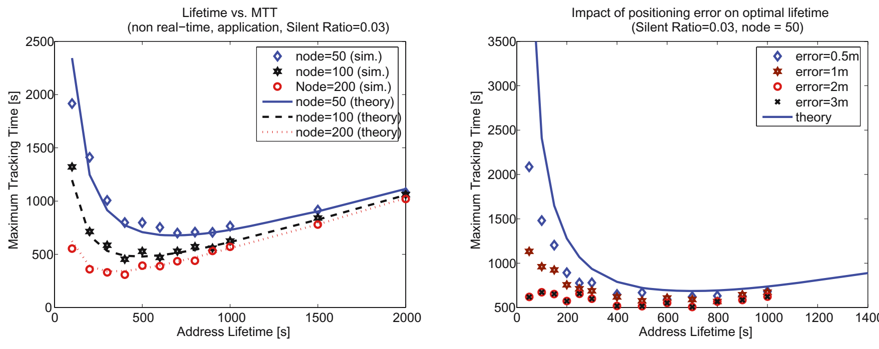


Fig. 6. (Left) Address lifetime vs. MTT under certain silent ratio, (right) Impact of estimation error on minimum MTT under certain silent ratio

Regarding the threshold time, We define the target size of silent cascade's GAS as S_t, and the length of cascade as l_0, threshold time can be derived as equation below:

$$TT(S_t) = (l_0 - 1) \times T_l + T_l/2 \begin{cases} l_0 = \log_2(S_t)/\log_2(s) & S_t > s_0 \\ l_0 = 1 & S_t \leq s_0 \end{cases} \quad (7)$$

From equation above, we can derive that the threshold time reaches minimum value when the target size of silent cascade's GAS is equal to that of a mix-zone's GAS.

In addition to the analytical study, we also use simulation to evaluate the performance of silent cascade given certain QoS and anonymity requirements. Left half of Fig.6 shows the relation between address lifetime and MTT when silent ratio is 0.03. There is not estimation error in the simulation results shown in this figure. Three lines indicate the theoretical value of MTT with different node density. Marks with three different shapes indicate corresponding simulation results. From this figure, we perceive that there is an optimal lifetime by which silent cascade achieves the same anonymity target within minimum duration of time. We also perceive the theoretical value derived from Eq.6 fits the simulation results quite well. However, we can observe difference between simulation and theoretical result increases when estimation error increases in right half of Fig.6. When estimation error is larger than some threshold (3.0 meter in the figure), there is no optimal lifetime at all. MTT is proportional to the length of silent period. We also observe the same trends in the simulation results regarding threshold time. Because of page limit, we omit its results here.

4.4 Analysis About the Upper Bound of System's Anonymity

In the analysis above, we assume that the inputs to different rounds of mix-network are independent, and there is not limits on the length of silent cascade. By using Eq.(2) and Eq.(3), we can easily derive that in principle the anonymity of a silent cascade can increases infinitely as the length of silent cascade increases.

However, we believe silent period with infinite anonymity does not exist because of following practical constraints and reasons.

The first reason is the limited address space. To prevent spoofing attacks on user's identifier, typically an identifier administrator prohibits multiple users to use the same identifier at nearby temporal-spatial space. As a result, if a group of nodes keep updating their addresses periodically, they may use up all available address at the end. Consequently, upper bound of anonymity of silent cascade is restricted by the size of address space. Given N bits of address space, and M users in the system, the maximum entropy of a silent cascade can be derived as below:

$$H(u_i) = \frac{2^N}{M} \times \log_2 M \tag{8}$$

The second issue is the duration of application. Maximum anonymity of silent cascade is limited by the duration of application.

The third issue is the correlation between information generated by the same station. When a station passes through a silent cascade, it may meet another station several times at different mix-zone. Because stations uses different pseudonym at the inputs of different mix-zones, we treat those inputs from the same station as independent inputs. According to the literatures[9], it is possible to find out the correlation between frames from the same station by its features such as transmission power or central frequency drift of RF antenna.

We make following assumption to analyze the upper bound of a silent cascade's anonymity due to correlated inputs. We assume there are $N + 1$ users including the target in the system. The target forms a chain of mix-zone when it moves within the system. We assume all mix-zones has same number of inputs S. Within each round of silent cascade, only those mixers, that has NOT met the target before in mix-zones, contribute to target's anonymity.

We define $p_{i,N}^m$ as the probability that target has met m different mixers from the beginning to the i-th round of silent cascade, and define $q_{i,N}^m$ as the probability that target meet m new users in i-th round of silent cascade. In addition, we also define $h(m_i)$ as the entropy of i-th mix-zone in the silent cascade, and H_i is the entropy of silent cascade, whose length is i.

In the first round of silent cascade, all mixers are new to target. Consequently, the following two equations hold.

$$p_{0,N}^j = q_{0,N}^j = \begin{cases} 1 & j = S \\ 0 & j \neq S \end{cases} \tag{9}$$

From the second round, we can derive their probability in $(i + 1)$-th stage based on the probability in i-th round as below.

$$p_{i+1,N}^k = \sum_{m=0}^{i\times S} q_{i,N}^m \times C_{N-m}^k \times C_m^{S-k} \tag{10}$$

$$q_{i+1,N}^k = \sum_{m=k-S}^{k} p_{i+1,N}^{k-m} \times q_{i,N}^m \tag{11}$$

According to Eq.(3), the entropy of a target' i-th mix-zone can be expressed as Eq.(12), and the entropy of silent cascade from beginning to i-th stage of silent cascade is Eq.(13).

$$h(m_i) = \log_2(\sum_{k=0}^{\min(i \times S, N)} (k+1) \times q_{i,N}^k) \tag{12}$$

$$H_i = \sum_{j=1}^{i} h_{m_j} = \sum_{j=1}^{i} (\log_2(\sum_{k=0}^{\min(j \times S, N)} (k+1) \times q_{j,N}^k)) \tag{13}$$

We also evaluate this upper bound by simulation. In the simulation, we define a new parameter correlation probability p_c as the probability that the inputs from the same station at different round of silent cascade adversary can successfully correlate. Fig.7 show the simulation result. From left half of the figure, we perceive that the entropy of silent cascade's GAS saturate when the delay of silent cascade increases. Furthermore as the size of a mix-zone's GAS increases, entropy of silent cascade's GAS saturates quicker. From right half of the figure, we also perceive that as the correlation probability increases, the upper limit of silent cascade decreases. However, we also perceive that pivot point where silent cascade begin to saturate is much larger than the practical range of silent cascade delay used location privacy protection. For example, the duration of time to achieve entropy of two is about 100 seconds, and the pivot point is about 2×10^5 seconds (55 hours) when the correlation probability is 100%. This means that although correlation of identifier restricts the upper bound of silent cascade's anonymity, its effect can be ignored in most of latest applications.

In summary of this section, Both analytical and simulation results indicates a clear trade-off relation between QoS (silent ratio or silent period) and delay of a silent cascade. Furthermore, it also shows that silent cascade is effective to provide enough location privacy without diminishing QoS at the expense of delay of silent cascade. Besides, for any given QoS and anonymity requirement, silent cascade has an optimal configuration regarding lifetime and silent period to achieve those requirements within minimum duration of time. Finally, analytical and simulation studies show the upper bound of silent cascade's anonymity and its impact on privacy protection protocol.

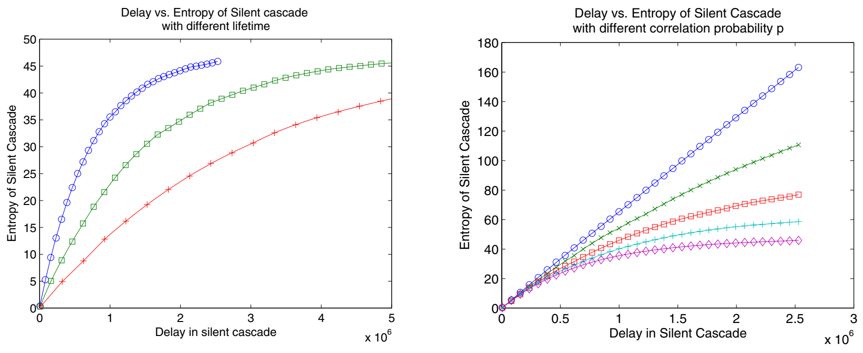


Fig. 7. Illustration of upper bound of anonymity due to correlation between inputs

5 Related Works

There are currently several research efforts to study methods to protect users' location privacy. Beresford and Stajano have proposed the concept of the *mix zone*[10] based on Chaum's mix network to protect location privacy for pervasive computing. They assumed LBS application providers are hostile adversaries, and suggested that application users hide their own identifier from providers. A mix zone for a group of users is defined as the largest connected spatial region in which none of the users in the area have registered an application callback. Gruteser and Grundwald [11] reduce the spatial-temporal resolution of location-based queries to guarantee a defined degree of anonymity in different locations. These mechanisms assume that location-based queries are generated so infrequently, that they can be viewed as independent queries (the adversary should be unable to link them to the same user). The time-series nature of a continuous stream of location information poses novel privacy challenges that have not been addressed. Gruteser and Hoh [12] described how trajectory-based linking (i.e. Multi Target Tracking) can infer a user's path from individual location samples provided by several users. In [13], Hoh and Gruteser present a path perturbation algorithm which can maximize users' location privacy given a quality of service constraint. In this paper, they propose a quality of service (QoS) metric in terms of the error in location samples imposed by tracking algorithms. In our previous works [1, 2], we proposed a method *silent period* to achieve the unlinkability between users' two or more locations by forming a *mix-zone* between nearby users. This proposal enhances user's location privacy at the expense of losing communication time.

6 Conclusion

In this paper, we proposed a scheme called silent cascade to enhance users' contradictory requirements on location privacy without diminishing communication QoS. We abstract the silent cascade protocol into a mix-network based formal model. The formal model reveals that silent cascade achieves anonymity without violating QoS by trading off the length of mix-network for anonymity. In addition, we derived optimal configuration of silent cascade to achieve target anonymity within minimum duration of time, and conclude the upper bound of silent cascade's anonymity and its impact on location privacy protection protocol.

References

1. Huang, L., Matsuura, K., Yamane, H., Sezaki, K.: Enhancing wireless location privacy using silent period. In: IEEE Wireless Communications and Networking Conference (WCNC 2005), NL, U.S. (2005)
2. Huang, L., Matsuura, K., Yamane, H., Sezaki, K.: Towards modeling wireless location privacy. In: Privacy Enhancing Technology(PET 2005), Cavtat, Croatia (2005)

3. Beresford, A., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing **2** (2003) 46–55
4. ITU-T: Methods for subjective determination of transmission quality, p.800 (1996)
5. Diaz, C., Serjantov, A.: Generalising mixes. In: Privacy Enhancing Technologies(PET2003). LNCS 2760, Dresden, Germany, Springer-Verlag (2003)
6. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Proc. Privacy Enhancing Technologies, Second International Workshop(PET 2002). Volume 2482 of LNCS., Springer (2002)
7. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Proc. Privacy Enhancing Technologies, Second International Workshop (PET 2002). Volume 2482 of LNCS., Springer (2002)
8. 3GPP: 1xev-dv evaluation methodology (v14), 3gpp2/tsg-c.r1002 (2003)
9. Castelluccia, C., Mutaf, P.: Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In: MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services, New York, NY, USA, ACM Press (2005) 51–64
10. Beresford, A.R.: Location privacy in ubiquitous computing. PhD thesis, University of Cambridge (2005)
11. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. of ACM MobiSys 2003, San Francisco, CA, USA, USENIX (2003) 31–42
12. Gruteser, M., Hoh, B.: On the anonymity of periodic location samples. In: 2nd International Conference on Security in Pervasive Computing(SPC2005). Volume 3450., Boppard, Germany, Springer (2005)
13. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: First International Conference on Security and Privacy for Emerging Areas in Communication Networks, Athens, Greece (2005)

Appendix

Table 2. Summary of the notations used in this paper

Name	Explaination	Name	Explaination
$p_{i,n}^{m}$	the probability that target met m *new* mixers in the i-th stage	$q_{i,n}^{m}$	the probability that target has met m mixers till i-th stage
C_{n}^{m}	Combination of n out of m	$USER(m_{i,j})$	users of mix mi, j
T_s	silent period	t_a	active period
T_l	lifetime	R_s	silent ratio, $R_s = T_s/T_l$
GAS	Geographical Anonymity Set	$s(m_{i,j})$	size of a mixe-zone $m_{i,j}$'s GAS
$h(m_{i,j})$	entropy of a mix-zone's GAS	$cascade(u_k)$	user u_k's silent cascade, consecutive list of mix-zone user u_k creates
$S(u_k)$	size of GAS of user u_k's silent cascade	MTT	Maximum Tracking Time (MTT) that a user can be tracked continuously
$TT(H_{target})$	duration of time for a user in silent cascade to achieve anonymity H_{target}	$delay(u_k)$	delay of user u_k's silent cascade
$length(u_k)$	number of mix-zone in user u_k's silent cascade		

Securing Information Gateways with Derivation-Constrained Access Control

Urs Hengartner[1] and Peter Steenkiste[2]

[1] University of Waterloo
uhengart@cs.uwaterloo.ca
[2] Carnegie Mellon University
prs@cs.cmu.edu

Abstract. In pervasive computing environments, information gateways derive specific information, such as a person's location, from raw data provided by a service, such as a videostream offered by a camera. Here, access control to confidential raw data provided by a service becomes difficult when a client does not have access rights to this data. For example, a client might have access to a person's location information, but not to the videostream from which a gateway derives this information. Simply granting access rights to a gateway will allow an intruder into the gateway to access any raw data that the gateway can access. We present the concept of derivation-constrained access control, which requires a gateway to prove to a service that the gateway needs requested raw data to answer a client's authorized request for derived information. Therefore, an intruder into the gateway will be limited in its capabilities. We provide a formal framework for derivation-constrained access control based on Lampson et al.'s "speaks-for" relationship. We demonstrate feasibility of our design with a sample implementation and a performance evaluation.

1 Introduction

Pervasive computing environments are full of services (e.g., sensors, databases, filesystems) that provide raw data. However, clients often do not access this raw data directly, instead they access it indirectly via an *information gateway* that extracts specific information from the raw data. For example, information gateways can derive a user's location from a videostream delivered by a camera or a user's current activity from her calendar. In short, a gateway accesses data offered by a service on behalf of a client.

Access control ensures that only authorized clients are granted access to confidential data. If clients access data via gateways, access control becomes difficult. On the one hand, a gateway needs access rights to the data in order to retrieve this data from a service. On the other hand, granting access rights to a gateway makes it vulnerable in case of an intrusion. Namely, an intruder into the gateway has access to any data that the gateway is authorized to get, and the intruder can actively issue requests for this data to services. Services will answer these requests if the gateway is authorized to access the requested data.

In this paper, we propose *derivation-constrained access control*. Here, before returning data to a gateway, a service requires the gateway to prove that the gateway is

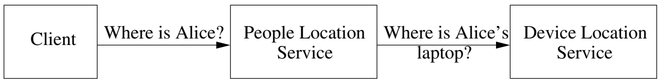


Fig. 1. Example gateway

asking for this data in order to answer a client's authorized request for information derived from this data. If there is no such request, an intruder into the gateway will not be able to prove that the gateway is asking for the data on a client's behalf. Therefore, the service will not grant the gateway (and the intruder) access to the requested data.

Let us illustrate derivation-constrained access control with an example application, which we will use throughout the paper. There is a service that locates people and a service that locates devices (see Figure 1). The latter service locates people indirectly by locating their devices, that is, the people location service acts as a gateway and contacts the device location service upon a client's request for a person's location. Assume that Bob asks the people location service for Alice's location information. The people location service then asks the device location service for the location of Alice's laptop. Derivation-constrained access control now makes the people location service prove to the device location service that the people location service is issuing this request in order to answer Bob's authorized request for Alice's location and that the location of Alice's laptop should be used for deriving Alice's location.

The main contribution of this paper is the concept of derivation-constrained access control. Three additional contributions are a formal model of this concept, a sample implementation, and a performance evaluation.

The rest of this paper is organized as follows: We present background material in Section 2. In Section 3, we discuss the concept of derivation-constrained access control in more detail. We introduce our formal model in Section 4 and apply it to an example scenario in Section 5. In Section 6, we present a security analysis. We discuss our sample implementation and its performance in Section 7 and 8, respectively. We comment on related work in Section 9.

2 Background

In this section, we discuss two types of gateways and contrast approaches chosen in related work with derivation-constrained access control. We also give our threat model.

The first type of gateways has the property that an authorized client has access rights both to the information offered by a gateway and to the data provided by the service that the gateway accesses on behalf of the client. For example, Alice has access both to her raw medical data, as collected by individual services (e.g., sensors), and to her health status, as derived by a gateway. In another example, she can access both her calendar data, as provided by a calendar application, and her location information, as derived from her calendar data by a gateway.

Lots of related work deals with this kind of gateways [1, 2, 3, 4, 5, 6]. In the proposed solutions, a client delegates its access right to a gateway, which enables the gateway to access the data at the service. To reduce the influence of an intruder into the gateway, related work suggests two approaches: First, instead of delegating all its access rights to a gateway, a client should delegate only the subset required by the gateway [3]. Second, a client should delegate only short-lived access rights [1, 2, 4]. Since the proposed solutions rely on the delegation of access rights, we use the term *delegation gateways* for referring to this kind of gateways.

In this paper, we focus on a second type of gateways, which has the property that an authorized client has access rights to the information offered by a gateway, but the client does not have access rights to the data offered by the service that the gateway accesses on behalf of the client. For example, a face detection application acts as a gateway and extracts somebody's location from a videostream. Even though Alice's location can be derived from a videostream and she can access her location information, she cannot access the videostream directly. In another example, Carol might have access to Bob's location information, but she does not have access to Bob's calendar data, from which a gateway derives his location. Finally, a gateway can derive the location of a person from the location of her laptop. A company might decide that, for administrative reasons, its employees should not have access rights to their laptop's location information. On the other hand, an employee should have access rights to her location information. We use the term *derivation gateways* for referring to this kind of gateways.

Related work has largely ignored derivation gateways. Delegation of access rights is not applicable here, since a client does not have access rights to the data offered by a service. Instead, derivation gateways are typically assumed to have separate access rights to data provided by a service. Therefore, access control ignores that this data is used by a gateway for deriving specific information, which makes the gateway vulnerable in case of an intrusion.

Derivation-constrained access control exploits such derivation properties. Namely, it makes a gateway prove to a service that the gateway is asking for data in order to answer a client's authorized request for derived information. The concept has the additional benefit that a gateway does not have to make access decisions. Only a service needs to make such decisions. Therefore, we can run access control in an end-to-end way.

We assume the following threat model: Gateways are not malicious to start with, but they become corrupted if an attacker breaks into them. In the case of corruption, an attacker has full control over a gateway and uses the gateway to contact other services. Corrupted gateways and clients that are authorized to get information offered by a gateway do not collude. Similarly, services and corrupted gateways do not collude.

3 Derivation-Constrained Access Control

In the rest of this paper, we use the term "information" for referring both to (raw) data, as offered by a service, and to derived information, as provided by a gateway.

Derivation-constrained access control requires two basic components. First, we need *derivation-constrained access rights*, which constrain access rights to an entity such that the entity has to prove that it is asking for the information listed in the access right in order to answer an authorized request for derived information. Second, in order to be able to identify derived information, we need formal *derivation properties* between information a gateway asks for and information a client asks for.

Instead of having two separate components, it is possible to directly include derivation properties in access rights. This approach has the drawback that it increases the number of access rights that need to be issued. However, it turns out that the approach restricts the flexibility of intruders that have broken into a gateway. We discuss this tradeoff in Section 6.

We now examine access rights and derivation properties in more detail and describe their application in derivation-constrained access control.

3.1 Derivation-Constrained Access Rights

Access rights to information are specified as follows:

[**Issuer** gives **subject** access to **information** under some **constraints**].

Issuers grant access by defining an access right.
Subjects are granted access rights. A subject can become an issuer by defining another access right and by granting the access right to another subject.[1] Therefore, there can be an entire chain of access rights delegating an access right. We call the first access right in the chain the *initial* access right.
Information describes information (e.g., a location or a videostream) about or from a specific entity (e.g., Alice or a camera in a building). For each type of information, there must be an *owner* who is allowed to define its initial access right. For example, Alice defines the initial access right for her location information and a building administrator defines the initial access right for a videostream provided by a camera in the building.
Constraints define under what conditions the access right is valid. There can be different types of constraints (e.g., time-based). In derivation-constrained access control, a constraint requires that the information in the access right is used for answering a request for derived information. We call access rights having such a constraint *derivation-constrained access rights*.

In our example application, Alice's company, as the owner of Alice's laptop, defines a derivation-constrained access right stating that the people location service can access the location of Alice's laptop.

3.2 Derivation Properties

We need derivation properties to tie information asked for by a gateway to information asked for by a client.

[**Issuer** says that **subject information** is derived from **issuer information**].

Issuers state derivation properties.
Issuer information is the information offered by a service. A gateway uses the issuer information to derive the subject information.
Subject information is the information offered by a gateway. A gateway derives the subject information from the issuer information.

[1] If we did not allow subjects to become issuers, they could proxy on behalf of other entities.

Only the owner of the issuer information and people having access to this information are allowed to specify a valid derivation property for this information.[2] The reason for this requirement is that a derivation property is used for controlling access to the issuer information. If a random client was able to define derivation properties, the client could define a derivation property with information owned by the client as subject information and send an authorized request for this subject information to a gateway. Assuming the gateway is granted access to the issuer information in a (derivation-constrained) access right, the gateway could then successfully query a service for this issuer information and leak the (derived) information to the client.

There is no need to specify derivation properties for each request issued by a client, they need to be defined only when there is a change in the derivation properties of information. Furthermore, the same issuer information can show up in multiple derivation properties. For example, Alice's calendar could be used for deriving both Alice's current location and her current activity. Similarly, subject information can occur in multiple derivation properties. For instance, Alice's location could be derived from Alice's calendar or from her laptop's location information.

Let us revisit our example application. Alice's company, as the owner of Alice's laptop, defines a derivation property that has the laptop's location information as issuer information and Alice's location information as subject information.

3.3 Access Control

Upon receiving a request for information, a service needs to make its access decision based on either access rights that are not derivation-constrained or derivation-constrained access rights and derivation properties. For both cases, the service needs to validate the access rights that grant access to the requesting gateway. Namely, the service must ensure that the information listed in the access right (or in a chain of access rights) corresponds to the information provided by the service. In particular, the access right and the service must agree on the owner of the information. For example, the administrator of a location service run by a company might decide that the company owns employee Alice's location information, as offered by this service. Therefore, the location service must not use access rights which state that Alice owns the information listed in the access right. In this case, the company is the issuer of the initial access right. On the other hand, the administrator of a location service in a mall can declare that Alice owns her location information, as provided by this service. Therefore, the service exploits access rights that list Alice as the owner of her location information. In this case, Alice is the issuer of the initial access right.

If the service makes an access decision based on derivation-constrained access rights and derivation properties, it will also have to validate the following properties:

- There must be a derivation property with the requested information as issuer information.
- There must be a request from a client to the gateway asking for the subject information in the derivation property.

[2] If we prevented people having access from issuing derivation properties, they could proxy.

– There must be a or a chain of access rights that grant the client access to this subject information.
– To avoid replay attacks, the client's request needs to be fresh.

In our example application, the device location service needs both Alice's derivation-constrained access right and the corresponding derivation property. In addition, it requires a fresh request from, for example, Bob asking for Alice's location information and an access right granting Bob access to this information.

4 Formal Model

In this section, we introduce a formal model for defining derivation-constrained access rights and derivation properties. The model allows us to formally reason about access control. It also provides the foundation for our implementation, which is based on digital certificates (Section 7). Our formal model requires a scheme for representing information in access rights and derivation properties. As explained in Section 3, for each type of information, there must be an owner. To simplify associating information with its owner, we make the owner part of the information representation scheme. Namely, we use $A.x$ for referring to information x with owner A. Example information is Alice.location_of_Alice or Building_Administrator.videostream_of_camera.

4.1 Conventional Access Control

We now revisit access control in distributed environments, as discussed by Lampson et al. [4] (and extended by Howell and Kotz [3]), and enhance their formal model to express the exact requirements that a client needs to fulfill in order to be granted access to information. In Section 4.2, this enhancement will allow us to easily add support for derivation-constrained access control to the existing access control model.

For expressing access rights to information, we exploit Howell and Kotz's "restricted speaks-for" relationship, which is based on Lampson et al.'s "speaks-for" relationship. We present the relationship based on an example. The relationship $B \overset{T}{\Longrightarrow} A$ denotes that B speaks for A regarding the statements in set T, that is, if A issues a statement that is in T, B also issues this statement. We are interested in statements expressing read access to information, for example, "read $D.x$". In the rest of this paper, we use the shortcut $B \overset{D.x}{\Longrightarrow} A$ instead of $B \overset{\{\text{“read } D.x\text{”}\}}{\Longrightarrow} A$. As we will see below, this speaks-for relationship grants B access to information $D.x$, assuming that A itself has access to this information.

According to the "handoff axiom" [4], the relationship $B \overset{D.x}{\Longrightarrow} A$ can be established by A. Formally, ($\supset$ denotes implication.)

$$\vdash (A\ \textbf{says}\ (B \overset{D.x}{\Longrightarrow} A)) \supset (B \overset{D.x}{\Longrightarrow} A). \tag{1}$$

A statement of the form A **says** $(B \overset{D.x}{\Longrightarrow} A)$ corresponds to an access right, as introduced in Section 3.1, except that there is no support for constraints. For example, Alice **says** (Bob $\overset{\text{Alice.location\_of\_Alice}}{\Longrightarrow}$ Alice denotes that Alice grants Bob access to her location information. The speaks-for relationship is transitive, which allows delegation of access rights.

When a service receives a request "read $D.x$" from principal C, it needs to ensure that C speaks for a principal on the service's access control list ("ACL") for this particular resource. Abadi et al. [7] introduce the construct D **controls** s to encode that principal D is on a service's ACL for resource s. The construct is defined as

$$D\ \textbf{controls}\ s \equiv ((D\ \textbf{says}\ s) \supset s).$$

As mentioned in Section 3, the owner D of information $D.x$ is responsible for defining the initial access right to $D.x$, formally, D **controls** "read $D.x$".

We can now summarize access control: Given D **controls** "read $D.x$" (i.e., a service's ACL), $C \xRightarrow{D.x} D$ (i.e., a speaks-for relationship derived from a or a chain of access rights), and C **says** "read $D.x$" (i.e., a request), a service concludes "read $D.x$" and grants C access to $D.x$.

For our purposes, the definition of the **controls** construct is not sufficient. It is important to guarantee freshness of a request in order to avoid replay attacks, as stated in Section 3.3. Strictly speaking, freshness demands that a request is new. We relax this requirement and demand only that a request is recent. As it turns out, Lampson et al.'s formal model easily supports this relaxed requirement, and the original requirement is difficult to achieve in our problem setting (see Section 7.2).

To formalize freshness, we observe that Lampson et al. suggest a statement form s **until** t to denote the lifetime t of statement s. We consider a statement fresh if its lifetime has not expired and extend Abadi et al.'s definition of ACLs to support freshness of a request: (**time** is a constant denoting the current time.)

$$D\ \textbf{controls}\ s \equiv (((D\ \textbf{says}\ s)\ \textbf{until}\ t) \wedge (t \geq \textbf{time}) \supset s).$$

4.2 Derivation-Constrained Access Control

We are now ready to formalize derivation-constrained access control, as discussed in Section 3, by extending the definition of D **controls** s .

We first introduce the convention to mark information in derivation-constrained access rights with the + sign (e.g., A **says** $(B \xRightarrow{D.x^{+}} A)$). This way, when a service detects marked information in a speaks-for relationship, it uses this relationship only for derivation-constrained access control, not for conventional access control.

We also introduce a notation for expressing derivation properties. The statement

$$E.y \mapsto D.x \tag{2}$$

denotes that subject information $E.y$ can be derived from issuer information $D.x$.

As explained in Section 3.2, we assume that for a derivation property to hold, the principal owning the issuer information or a principal speaking for the owner needs to issue the property, that is,

$$\vdash (F\ \textbf{says}\ (E.y \mapsto D.x)) \wedge (F \xRightarrow{D.x} D) \supset (E.y \mapsto D.x). \tag{3}$$

We can now formulate derivation-constrained access control. In particular, we have a service use a different definition for D **controls** s if s involves information marked with the + sign. Formally,

$$\begin{aligned} & D \textbf{ controls } \text{“read } D.x^{+}\text{”} \\ & \equiv (((D \textbf{ says } \text{“read } D.x^{+}\text{”}) \textbf{ until } t_1) \wedge (t_1 \geq \textbf{time}) \\ & \wedge (E.y \mapsto D.x) \\ & \wedge ((E \textbf{ says } \text{“read } E.y\text{”}) \textbf{ until } t_2) \wedge (t_2 \geq \textbf{time}) \\ & \supset \text{“read } D.x^{+}\text{”}). \end{aligned} \tag{4}$$

The first condition is straightforward: There must be a fresh and authorized request, “read $D.x^{+}$”. In the second condition, we require that there is information $E.y$ that can be derived from $D.x$.[3] The third condition calls for a fresh and authorized request for the derived information, $E.y$.

Note that the third condition can be replaced by a condition requiring a request for $E.y^{+}$ and conditions requiring the existence of another derivation property and of another request. This property makes deployment of our mechanism feasible in scenarios where a client's request for information is dealt with by multiple, cascaded gateways.

5 Example Application

In this section, we demonstrate a sample application of derivation-constrained access control. We assume that individuals and gateways or services each own a public and a private key. We use the public key for identifying entities. For example, the notation K_{Alice} denotes Alice's public key. Entities use the private key for signing statements.

Assume that a people location service, PL, derives Alice's location information, K_{Alice}.loc_Alice, from her laptop's location information, K_{ACME}.loc_Alice_laptop, where the laptop is owned by Alice's company, ACME, and its location information is offered by a device location service. Note that the two types of information are owned by two different entities. We show the relevant statements in Table 1.[4] We start with statements that individuals make before an actual request is issued. Alice grants Bob access to her location information (S1). We expect this statement to be long lived. ACME grants the people location service access to Alice's laptop's location information in a derivation-constrained access right (S2), again in a long-lived statement. ACME also states that the laptop's location information can be used for deriving Alice's location information (S3). Finally, the device location service has ACME on its ACL for the laptop's location information (S4).

Bob sends a request for Alice's location information to the people location service (S5) and issues a short-lived access right for Alice's location information to the people location service quoting (denoted by “|”) Bob (S6). This way, the people location service must quote Bob when it wants to use this access right. As observed by

[3] Derivation properties can also have lifetimes; we leave them away for readability reasons.

[4] In a complete approach, we would also have to take communication channels used by principals into account, but we ignore them for simplicity reasons. Similarly, we assume that the axioms introduced by Lampson et al. [4] have been modified to take lifetimes into account.

Table 1. Statements in the people location scenario. (S1)-(S4) are established during setup, (S5)-(S16) are established during access control.

Step	Statements
(S1)	$(K_{\text{Alice}}$ **says** $K_{\text{Bob}} \xRightarrow{K_{\text{Alice}}.\text{loc\_Alice}} K_{\text{Alice}})$ **until** t_1
(S2)	$(K_{\text{ACME}}$ **says** $K_{\text{PL}} \xRightarrow{K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+} K_{\text{ACME}})$ **until** t_2
(S3)	K_{ACME} **says** K_{Alice}.loc_Alice $\mapsto K_{\text{ACME}}$.loc_Alice_laptop
(S4)	K_{ACME} **controls** $K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+$
(S5)	K_{Bob} **says** "read K_{Alice}.loc_Alice"
(S6)	$(K_{\text{Bob}}$ **says** $K_{\text{PL}}\|K_{\text{Bob}} \xRightarrow{K_{\text{Alice}}.\text{loc\_Alice}} K_{\text{Bob}})$ **until** t_3
(S7)	$K_{\text{PL}}\|K_{\text{Bob}}$ **says** "read K_{Alice}.loc_Alice"
(S8)	K_{PL} **says** "read $K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+$"
(S9)	$(K_{\text{PL}} \xRightarrow{K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+} K_{\text{ACME}})$ **until** t_2
(S10)	$(K_{\text{ACME}}$ **says** "read $K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+$") **until** t_2
(S11)	K_{Alice}.loc_Alice $\mapsto K_{\text{ACME}}$.loc_Alice_laptop
(S12)	$(K_{\text{PL}}\|K_{\text{Bob}} \xRightarrow{K_{\text{Alice}}.\text{loc\_Alice}} K_{\text{Bob}})$ **until** t_3
(S13)	$(K_{\text{Bob}}$ **says** "read K_{Alice}.loc_Alice") **until** t_3
(S14)	$(K_{\text{Bob}} \xRightarrow{K_{\text{Alice}}.\text{loc\_Alice}} K_{\text{Alice}})$ **until** t_1
(S15)	$(K_{\text{Alice}}$ **says** "read K_{Alice}.loc_Alice") **until** t_3
(S16)	"read $K_{\text{ACME}}.\text{loc\_Alice\_laptop}^+$"

Howell and Kotz [8], this quoting prevents a gateway, like the people location service, from having to make an access decision itself. Instead, by quoting a client, the gateway notifies a service of the identity of the client, which allows the service to make an access decision in an end-to-end way. Statement (S7) shows an example of quoting: The people location service forwards Bob's request to the device location service by quoting Bob. The people location service also issues a request for the location of Alice's laptop to the device location service (S8).

The device location service validates the authenticity of Statement (S2) using Axiom (1) and deduces that the people location service can speak for ACME on behalf of Alice's laptop (S9). Therefore, ACME supports the request of the people location service (S8) for the laptop's location information (S10). The device location service also validates Statement (S3) using Axiom (3) and deduces that the derivation property is valid (S11). Based on the people location service's access right (S6), the device location service concludes that the people location service (quoting Bob) can speak for Bob (S12) and thereby Bob supports (S13) the request of the people location service (S7). After validating that Bob can speak for Alice (S14) based on his access right (S1), the service concludes that Alice also supports the request (S15). (We assume that $t_3 \ll t_1$ since Statement (S1) is long lived, whereas Statement (S6) is short-lived.) Given the ACL for the laptop's location information (S4) and Definition (4), the device location service uses Statements (S10), (S11), and (S15) to deduce that the people location service should be granted access (S16) (assuming $t_2 \geq \mathbf{time}$ and $t_3 \geq \mathbf{time}$).

Note that among the statements in Table 1, only Statements (S1)-(S6) need to be specified by individuals, all the other statements can be derived or specified automatically by the access control framework (see Section 7). Furthermore, only two of these statements, (S5) and (S6), need to be issued for each request. (In Section 7.2, we present an optimization that combines these two statements in a single statement.) Statements (S1)-(S4) need to be re-issued only when they expire or when there is a change in the derivation properties of information. Among these four statements, Statement (S3) is the only additional statement that is required by derivation-constrained access control, the other three statements are also required in a traditional access control scenario, where the people location service is granted unrestricted access to the laptop location information. (In addition, this scenario would require a statement denoting the ACL of the people location service because this service would have to run access control.)

6 Security Analysis

Since our formal model is based on Lampson et al.'s model, we inherit its properties. For instance, the model does not prevent a principal having an access right from delegating this access right to other principals. For derivation-constrained access control, this principle implies that an intruder breaking into a gateway can delegate the gateway's (derivation-constrained) access rights to other, potentially corrupted gateways. While this attack will not grant the first gateway access to additional information, the other gateways could exploit the delegated access rights if there was an authorized request from a client to them.

As mentioned in Section 3, we keep access rights and derivation properties separate. A disadvantage of this approach is that intruders into a gateway get more flexibility. Assume that there are multiple derivation properties, all of them having the same issuer information. For example, multiple kinds of information can be derived from someone's calendar. The owner of the calendar grants derivation-constrained access to a gateway because the owner expects the gateway to provide one particular derivation. (For example, the owner expects the gateway to provide location information.) If an authorized client now asks the gateway for subject information listed in any of the derivation properties, the gateway will be granted access, even though the owner does not expect the gateway to perform the derivation listed in the derivation property exploited by the gateway. Therefore, an intruder into a gateway can increase its chances of success by trying to trick authorized clients into contacting the corrupted gateway for any of the different types of subject information. There are different ways to address this attack: We can instruct clients which gateways to use for retrieving a particular kind of information, for example, when a client is granted an access right to this information. We can also avoid the attack by having the owner of the issuer information combine the derivation-constrained access right granted to the gateway and the derivation property targeted at this gateway in a single statement such that an attacker cannot exploit them separately. However, this approach could require the owner of the issuer information to issue more access rights. Namely, if a gateway extracted multiple information items from the issuer information (e.g., multiple people's location could be derived from a camera picture), the gateway will need separate access rights for each information item. When derivation properties are separate, the gateway requires only one access right.

Derivation-constrained access control mainly targets services that provide dynamic information (e.g., location information). An intruder into a gateway gets to see information retrieved by the gateway upon an authorized request from a client. Therefore, if this information was static, an attacker could just wait for an authorized request to occur, and there is less incentive for the intruder to issue its own requests in the meantime.

7 Implementation

Let us discuss our sample implementation of derivation-constrained access control.

7.1 Access Control

When receiving a request asking for access to information $D.x$ or $D.x^+$, a service S needs to establish "read $D.x$" or "read $D.x^+$", based on Definitions (2) or (4). There are multiple ways to establish these statements. A service can retrieve a set of access rights and derivation properties, locate relevant statements in this set, and validate them. However, this approach puts a potentially heavy load on the service. It is really only the validation step that the service has to perform itself, whereas it is possible to offload the first two steps to some other entity. The validation step tends to be cheap (see Section 8), which makes it possible for resource-limited services (e.g., a sensor) to run this step. In our implementation, we make the client and the gateway gather any statements required for being granted access and have them ship these statements, together with the request, to the gateway and the service, respectively. Resource-limited clients can further offload the gathering of statements, if needed. We implemented derivation-constrained access control within an existing framework for such *proof-based* access control [8].

7.2 Statements

To express the access rights and derivation properties introduced in Section 4, we rely on digital certificates. Digital certificates are signed, which makes it easy to identify the issuer of a statement. Our framework is based on SPKI/SDSI certificates [9]. The existing SPKI/SDSI specification supports the inclusion of constraints in an access right. However, interpretation of a constraint is left to an application. Therefore, we extended the specification and let derivation-constrained access rights and derivation properties become first-class citizens. This way, we can support derivation-constrained access control directly in the access control framework.

As shown in Section 5, a client that issues a request also needs to generate a short-lived access right for the information in the request to the gateway. If there are multiple, cascaded gateways, each will have to issue short-lived access rights to the next gateway in the chain. As we will see in Section 8.1, generating a digital signature covering such an access right is expensive. In our implementation, we reduce cost by not making entities issue short-lived access rights. Instead, we have the original issuer of a request sign this request, where the lifetime of this signature is short. Gateways just forward this signed request. We assume that any entity that is in the possession of a signed request is implicitly granted access to the information in the request. Formally, this entity can speak for the original issuer of the request on behalf of the information in the request. To be secure, this optimization requires two precautions: First, we have to ensure that attackers cannot snoop traffic and obtain access rights by observing signed requests. Therefore, communication has to be confidential. Note that confidentiality is required anyway because we exchange confidential information (such as a person's location information). Second, a service can use a speaks-for relationship that is implicitly derived from a signed request only for running access control for this particular request; it must never reuse such a relationship for other requests. The latter precaution is required since gateways do not run access control. Without it, an attacker could contact a gateway that previously obtained information from a service and have it contact the service again. If the service reused the previously derived speaks-for relationship, it would grant the gateway access again (unless the statement has expired).

To support the **until** construct, we exploit that SPKI/SDSI certificates can have a lifetime associated with them and have the issuer of a statement assign a lifetime to the statement. We use lifetime to achieve freshness. We could also integrate a nonce-based approach, where the entity that wants to issue a request to a target receives a nonce from the target and returns this nonce in its request. However, our communication pattern, where a client interacts only with the gateway, but not with the service, does not allow for nonces. There is no way for the service to give a nonce to the client to prove freshness of the client's request.

7.3 Execution Environment

We use the Aura ubiquitous computing environment [10] as our testbed for the implementation and deployment of derivation-constrained access control. Entities within this environment communicate with each other using a protocol that exchanges XML-encoded messages embedded in HTTP requests/responses. We extended this protocol to support the transmission of proofs of access, as explained in Section 7.1. SSL gives us peer authentication and message confidentiality and integrity.

8 Evaluation

We present a measurement-based analysis of derivation-constrained access control and discuss the results.

8.1 Measurements

We measure the time it takes for a gateway and a service to process a request, where processing includes access control. Our scenario roughly corresponds to the example application from Section 5. We assume that Alice grants Bob access to her location information. Bob contacts a service providing people location information. This service acts as a gateway and locates people by locating their devices. Namely, it contacts a service providing location information about badges worn by people. Alice, as the owner of the badge, specifies a derivation property between her badge's location information and her location information. She also issues a derivation-constrained access access right granting the people location service access to the badge's location information.

Our experiments run on a Pentium IV/2.5 GHz with 1.5 GB of memory, Linux 2.4.20, and Java 1.4.2. The public and private key operations operate on RSA keys with a size of 1024 bit. All experiments are run 100 times.

Table 2. Mean and standard deviation of elapsed time for security-related operations (in bold) and other expensive operations [ms]

Entity	Step	μ (σ)
Client	**Request generation**	27 (3)
Client	**SSL sockets creation**	52 (4)
Gateway	**Access control**	2 (1)
Gateway	Gather badge information	26 (3)
Gateway	**Request generation**	41 (3)
Gateway	**SSL sockets creation**	52 (5)
Service	**Access control**	3 (2)
Service	Gather location information	49 (30)
	Total	310 (37)

It takes about 310 ms for the client to send a request to the gateway and to receive a response. We present more detailed results in Table 2. The terms "client", "gateway", and "service" represent Bob, the people location service, and the badge location service, respectively. The most expensive operations are the two SSL sockets creations, one pair for the client-gateway connection and another pair for the gateway-service connection. For opening an SSL connection, the two peers need to authenticate each other. Each authentication requires an RSA signing or decryption operation operation, which takes about 17 ms. RSA is also responsible for most of the cost of generating a request, because a request needs to be signed. In our current implementation, any entity issuing a request needs to sign it. This approach supports scenarios, where there are multiple, cascaded gateways. If an entity knew that it is directly contacting a service, it could omit this signature. For example, the people location service would not have to sign the request for the badge's location information.

Compared to the cost for creating SSL sockets and generating a request, access control is cheap. The main cost is signature verification, which takes about 1 ms per signature. The gateway validates the signatures of the client's request and of the access right granting access to the client. Strictly speaking, derivation-constrained access control does not require a gateway to make an access decision. However, we have the gateway make such decisions, since they are cheap and can help against denial-of-service attacks, where clients send unauthorized requests to a gateway. The service needs to make an access decision. It validates the signatures of the client's request, of the access right granting access to the client, of the access right granting derivation-constrained access to the gateway, and of the derivation property. Overall, it has to validate four signatures.

Let us compare the cost of derivation-constrained access control to the cost of traditional access control, where the owner of the information provided by a service issues a long-lived access right to a gateway. This approach is less expensive than our approach. There is no need for the client to sign its request to the gateway (i.e., issue a short-lived access right to the gateway) and for the service to validate this access right and the access right of the client. However, long-lived access rights make the gateway vulnerable in case of an intrusion. Furthermore, derivation-constrained access control does not require a gateway to make an access decision, whereas the alternative approach does.

9 Related Work

Lots of related work deals with delegation gateways [1, 2, 3, 4, 5, 6]. We have discussed the limitations of these schemes in Section 2 and provide a more detailed discussion of this work in the first author's Ph.D. thesis [11, Chapter 4].

Derivation-constrained access control is based on the concept of rights amplification (also called privilege escalation). This concept has been used in other access control models. In Bertino et al.'s model [12], an access right becomes valid whenever an access right that it depends on becomes valid. In our scenario, the former access right could be a gateway's access right to information offered by a service and the latter one a client's access right to information provided by the gateway. However, the presented model applies only to a centralized environment, in which there is an administrator that manages both access rights and dependencies between access rights. In pervasive computing, individuals issue access rights and establish derivation properties. Some operating systems based on capabilities (e.g., Hydra [13]) also support rights amplification. Again, these access control models target a centralized environment. Jajodia et al. [14] present several possibilities for deriving access rights, but their model misses a temporal component and applies only to centralized environments.

In derivation-constrained access control, an intruder into a gateway gets to see any information given to the gateway when a service answers an authorized request from the gateway. We could avoid this information leak by having the service encrypt information such that only the client can decrypt it and by having the gateway compute on the encrypted information. This approach obviously worked if the gateway did not need to perform any computations on the information (e.g., when deriving the location of a person from the location of her laptop). The approach is also applicable to some very specific scenarios, where the gateway does have to run some computations on the encrypted information [15]. However, it is unclear how practical this approach is in general.

To be sure that our framework performs correctly, we need to prove that the access control logic is sound by incorporating our extensions into Abadi et al.'s semantic model [7]. (Alternatively, we could prove all our axioms as theorems in a logic known to be sound (e.g., Appel and Felten's logic [16]).) This is ongoing work.

10 Conclusions and Future Work

We introduced the concept of derivation-constrained access control, which allows one to limit the capabilities of an intruder into a gateway. We presented a notation for expressing derivation-constrained access rights to information and derivation properties between information, and we proposed a mechanism for exploiting both of them in derivation-constrained access control. The sample implementation and its deployment demonstrate the feasibility of our design. The performance of access control is competitive. A large fraction of the cost is caused by expensive, but required operations for authenticating peers.

We are deploying derivation-constrained access control in additional services, which provide other information than location information (e.g., task information).

Acknowledgments

We thank the anonymous reviewers for their comments. This research was supported by the Army Research Office through grant number DAAD19-02-1-0389 and by the NSF under award number CNS-0411116.

References

1. Gasser, M., McDermott, E.: An Architecture for Practical Delegation in a Distributed System. In: Proceedings of IEEE Symposium on Security and Privacy. (1990) 20–30
2. Kornievskaia, O., Honeyman, P., Doster, B., Coffman, K.: Kerberized Credential Translation: A Solution to Web Access Control. In: Proceedings of 10th Usenix Security Symposium. (2001)
3. Howell, J., Kotz, D.: A Formal Semantics for SPKI. In: Proceedings of 6th European Symposium on Research in Computer Security (ESORICS 2000). (2000) 140–158
4. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in Distributed Systems: Theory and Practice. ACM Transactions on Computer Systems **10**(4) (1992) 263–310
5. Neuman, B.: Proxy-Based Authorization and Accounting for Distributed Systems. In: Proceedings of International Conference on Distributed Computing Systems. (1993) 283–291
6. Sollins, K.R.: Cascaded Authentication. In: Proceedings of IEEE Symposium on Security and Privacy. (1988) 156–163
7. Abadi, M., Burrows, M., Lampson, B.: A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems **15**(4) (1993) 706–734
8. Howell, J., Kotz, D.: End-to-end authorization. In: Proceedings of 4th Symposium on Operating System Design & Implementation (OSDI 2000). (2000) 151–164
9. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. RFC 2693 (1999)
10. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project Aura: Towards Distraction-Free Pervasive Computing. IEEE Pervasive Computing **1**(2) (2002) 22–31
11. Hengartner, U.: Access Control to Information in Pervasive Computing Environments. PhD thesis, Computer Science Department, Carnegie Mellon University (2005) Available as Technical Report CMU-CS-05-160.
12. Bertino, E., Bettini, C., Samarati, P.: A Temporal Authorization Model. In: Proceedings of 2nd ACM Conference on Computer and Communications Security (CCS 1994). (1994) 126–135
13. Cohen, E., Jefferson, D.: Protection in the Hydra Operating System. In: Proceedings of 5th ACM Symposium on Operating Systems Principles. (1975) 141–160
14. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmaninan, V.S.: Flexible Support for Multiple Access Control Policies. ACM Transactions on Database Systems **26**(2) (2001) 214–260
15. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy. (2000)
16. Appel, A.W., Felten, E.W.: Proof-Carrying Authentication. In: Proceedings of 6th ACM Conference on Computer and Communications Security. (1999)

Information Flow Control to Secure Dynamic Web Service Composition*

Dieter Hutter and Melanie Volkamer

German Research Center for Artificial Intelligence (DFKI GmbH),
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{hutter, volkamer}@dfki.de

Abstract. The vision of a landscape of heterogeneous web services deployed as encapsulated business software assets in the Internet is currently becoming a reality as part of the Semantic Web. When pro-active agents handle the context-aware discovery, acquisition, composition, and management of application services and data, ensuring the security of customers' data becomes a principle task. To dynamically compose its offered service, an agent has to process and spread confidential data to other web services demanding the required degree of security. In this paper we propose a methodology based on type-based information flow to control the security of dynamically computed data and their proliferation to other web services.

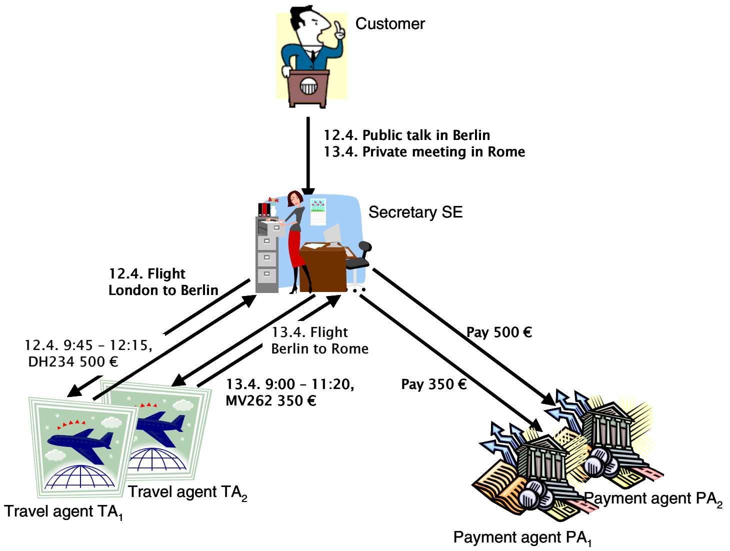
1 Introduction

The proliferation of web services as self-contained web accessible programs and the idea of the Semantic Web of making information *computer-interpretable* enables the dynamic composition of complex services assembled from various individual services and typically distributed over the web. Following the paradigm of pervasive computing, pro-active agents are installed on mobile phones or PDAs operating on the web and handle the context-aware discovery of appropriate services and use AI-based planning techniques to dynamically compose the retrieved service to solve complex tasks. Web services provide formal specifications of their well-defined semantics using languages based on description logics like OWL-S [19] or its predecessor DAML-S [7]. Based on these semantically annotated web services users who want to achieve a specific goal could be assisted by intelligent agents that automatically identify and compose the necessary services.

The introduction of web services in general and dynamic web service composition (e.g. [6, 24, 31, 24, 24, 14]) in particular requires appropriate security facilities to guarantee the security requirements of all participants. Web services have to be protected against misuse of their resources on the one hand, and on the other hand the customers of web services require guarantees with respect to the security (e.g. confidentiality or integrity) of their data. Security policies are used

* Parts of this work were sponsored by grants from the German Ministry for Technology and Education (BMBF) and the German Science Foundation (DFG).

to specify the security requirements of a system. The formalization of security policies is often guided by the idea to protect integrity or confidentiality by controlling the access to locations in which confidential information is stored. This notion of *access control* is typical for most of the approaches concerned with the specification of appropriate security policies for web services (e.g. [20, 8]). This results in a web service centered view of security. These policies describe access rights to web services and the delegation and withdrawal of these access rights. However, they are less suited to formulate the security needs of a customer, i.e. to formalize how web services have to deal with provided customer's information. In contrast, information flow control [17, 32, 16] relies on the idea of modeling confidentiality of data as restrictions on the flow of information between domains of a (global) system. Starting with the work of Goguen and Meseguer [9, 10], the restrictions on information flow have been formalized as independence properties between actions and observations of domains. In this paper we present an approach which uses information flow control techniques to control the distribution of customer's information in web services using security policies formulated and provided by the customer.


Customer
12.4. Public talk in Berlin
13.4. Private meeting in Rome
Secretary SE
12.4. Flight London to Berlin
12.4. 9:45 - 12:15, DH234 500 €
13.4. Flight Berlin to Rome
13.4. 9:00 - 11:20, MV262 350 €
Pay 500 €
Pay 350 €
Travel agent TA$_1$
Travel agent TA$_2$
Payment agent PA$_1$
Payment agent PA$_2$

A Simple Example. As an example consider a future-oriented politician who uses intelligent web services to organize his traveling in Europe. Living in London, he has to give a public talk in Berlin and he will meet the Italian prime minister the day after in a secret get-together. He informs his web service 'secretary' implemented on his PDA about his intentions. This web service decomposes the problem into organizing the flights between London, Berlin and Rome and making arrangements for payment. The secretary contacts a travel agency for the desired flights and obtains in return specific flight information and the corresponding prices. Afterward, the secretary selects a payment service which performs the payment of the selected flights.

This example illustrates how the confidentiality of the get-together in Rome enforces the confidentiality of information dynamically calculated by various web services when planning the traveling. To keep the get-together secret, the flight to Rome and the information provided by the travel agent about this flight have to be confidential. If the price of the flight depends on the flight information then also the price has to be confidential. This means that even if the payment service does not obtain any details of the booked flights, the politician has to trust or ensure that the payment service keeps the possible deduced information about his whereabouts confidential. Otherwise one might deduce flight information from the amount of money the service has to pay. The situation changes if the travel agency offers a flat rate for European flights. Then the price of the flights does not depend on the selected destinations and its calculation is independent of specific flight information. In this case the politician does not have to trust the discretion of the payment service with respect to his whereabouts. Furthermore, while our politician trusts his payment service with respect to financial information he may have made bad experiences with payment systems offered by some travel agencies and instructs his payment system not to pay by credit card any service offered by these agencies.

Summing up, our politician has his individual security policy with respect to the confidentiality of his whereabouts and his bank and credit card information. This policy has to be dynamically extended to cope with newly computed or generated information and also with subordinate web services not primarily known to our politician.

Access Control vs. Information Flow. Standard approaches (like, for instance, REI [20], Ponder [8]) are based on access control policies that control the execution of actions on individual objects. However, access control policies suffer from the problem of Trojan Horses or other information leakage using hidden channels. The reason for this is that no control is enforced on the use of the provided data once it is released to an authorized web service. Improper processing of confidential information and calls to subsiderary web services can disseminate secrets to web services that are not authorized to read this information. Mandatory access control might overcome some of these problems by labeling each information and service with some security level. Policies similar to Bell/LaPadula [2] could be used to control read/write operations on data depending on the labeling of data and web services. But still we are faced with two major problems:

First, there is no central authority in the web that is able to fix the security labels of all services and data. Both customers and web services will provide information to accomplish a requested service, and both sides will have individual perceptions about how the provided data may be used. Similar to the dynamic composition of web services there is a need for a dynamic and consistent composition of the related security policies of all participants.

Second, rather than providing statically stored data, web services will search for new ways to *compute* requested information from various data available on the net. The dynamic composition of web service results not only in a dynamic synthesis of programs (actions) for processing the data but also in a dynamic generation of new types of data which have to be dynamically classified according to their stored information.

In this paper we adopt techniques from language-based information flow control to control the secrecy of dynamically created data according to the policies of the involved web services. Each data is equipped with a type specifying its classification with respect to different (user-defined) security categories. In our example such a type would encompass for instance the degree of privacy of the whereabouts and the degree of confidentiality of payment information. Customers formulate their security requirements by attaching types to customer data and to individual web services or classes of web services.

The effects of this typing are twofold. First, web services can only be used for service composition if they provide the necessary clearance for the information they would receive when executing the service. Second, the computation of any low-security data has always to be independent of high-security data. Once a new data is synthesized, it is automatically classified according to the classifications of the data that have been used to compute it.

We start with a closer look on our approach in Section 2. In Section 3 we describe how to generate and specify security policies in our approach. Section 4 introduces a type calculus which is used to compute the security classification of both, newly computed data and composed services. We finish our paper with a comparison of our work with existing approaches in Section 5.

2 Dynamic Web Service Composition

Web services are software components distributed on various hosts and communicating over the web using standard protocols based on XML. Internet description languages are typically used to describe the interfaces. Following the idea of the semantic web [4], the idea of dynamic web service composition (cf. e.g. [30, 3]) is that web services provide their semantics as formal specifications such that other web services incorporating AI-planning algorithms can make use of these web services to solve complex (previously unknown) tasks. Different planning algorithms are proposed to implement dynamic web service compositions (cf. [21] for a survey). For our purposes we are not interested in the concrete planning process but we will operate on (partial) results of such planning processes just before the synthesized plans (or parts thereof) are executed. We consider the result of the planning process as (partial) programs formalized in some "standard" sequential programming language. In general they contain calls to other web services that are basically considered as procedure calls. Data is provided to web services via input parameters and returned by the web service via output parameters.

We distinguish between *atomic* and *composed* web services. While an atomic web service provides the service defined in its formal specification without requesting other web services, a composed web services will distribute subproblems to other web services. The execution of a web service can be considered as a tree: all leaves are atomic web services while inner nodes are compound web services. Plans have to be synthesized for each individual service request because in the setting of agent-based provision of web services there is no static global system but instead it is constantly changing by accretion and leave of individual agents and web services.

In order to satisfy a requested service, a web service receives sensitive data together with a security policy which is basically formulated by the customer and which guides the usage of data with respect to other web services. On the one hand it classifies the privacy of the provided data according to different information classes (like for instance traveling or payment information). For example, our politician rates his get-together in Rome as confidential while his talk in Berlin is public. On the other hand the security policy formalizes the trust a customer has in individual web services or families of web services by assigning corresponding clearances (split into the different classes of information) to web services. Then before actually calling a web service, it has to be checked that its clearance is sufficiently high to obtain the classified data. The called web service inherits the customer's security policy and extends it with respect to the clearance of previously unknown web services or the classification of data computed by the web service. A web service is not allowed to change the classification of provided data. Downgrading sensitive data would violate the security requirements of the customer but also upgrading data is typically useless because the data could have been already disseminated to web services which do not possess the newly required clearances.

Web services implement a bi-directed communication, which is important for policy descriptions: The web service receives a request including sensitive data from its caller and sends data back, which may also be sensitive. Hence, web services have to classify their newly computed data according to the security policies of web services and customers providing the processed information. A web service implementing an interface to a database may provide, for instance, its data only to web services with special clearances. Web services that process only data provided by the customer have to classify the computed data according to the classification of the used customer's data.

3 Security Policies for Dynamic Composition

As mentioned in the introduction we adopt the notion of non-interference to formulate confidentiality as independence between data. The low-security data must not depend on any high-security data. As a consequence, actions affecting low-security data are only allowed if they only access low-security data. More generally, the classification of any computed/synthesized information has to be at least as high as the classifications of all used data; i.e. no secret bit of information must be disclosed in public information. Since the way information is assembled is dynamically planned in web services, also the classification of information has to be done dynamically during the plan construction and execution. In the following we adopt a type calculus developed by Volpano and Smith (e.g. [28, 29, 25]) to encode classifications as types and use rules of the type calculus to propagate the types of data along the composition of a given plan.

The classification of data determines the accessibility of this information by individual web services. In order to receive classified information, web services must provide a clearance that is at least as high as the classification of the data. However, in contrast to mandatory access control, there is no central authority that assigns clearances to individual web services but each individual customer has its own perception about the security categorization of individual or families of web services. Thus besides the classification of provided information, a customer has also to provide rules that allow invoked web services to assess the clearance of potential subcontractors and to decide whether this clearance is sufficiently high to deal with the data required to perform the service.

Clearances and classifications are formalized with the help of standard information flow policies. Such a policy is a lattice $(SC, \leq)$ where SC is a finite set of *security classes* that is partially ordered by $\leq$. In its simplest form SC might be a set containing two elements, e.g. H and L denoting secret and public, but we can easily encode also integrity classes (cf. [29] for an example) to differentiate between trusted or untrusted information.

As illustrated in our example we want to subdivide the clearance of a web service according to different types of information. For instance, we may want to classify a travel agency as being high with respect to our whereabouts but low with respect to our financial situation. Therefore, we combine different information flow policies $(SC_1, \leq_1), \ldots, (SC_n, \leq_n)$ to a *composed* flow policy $(SC, \leq)$ by $SC = SC_1 \times \ldots \times SC_n$ and $\langle \tau_1, \ldots, \tau_n \rangle \leq \langle \tau'_1, \ldots, \tau'_n \rangle$ iff $\tau_i \leq \tau'_i$ holds for all $1 \leq k \leq n$. Least upper bound (and greatest lower bound, respectively) are computed by the least upper bounds (and greatest lower bounds, respectively) of each component.

Once a customer charges a web service with some task he provides (partially) classified information to the web service. For instance, our politician tells his secretary about his confidential gathering in Rome and that it has to be kept secret to some extent. The potential distribution of this data to other web services is regulated by (i) the classification of the provided data (being, for instance, secret) and (ii) the clearances the customer assigns to the web services with respect to this type of information. Let $(SC, \leq)$ be a (compound) information flow policy, then we call a partial mapping σ_{ws} from the set of web services to SC a *web service clearing* wrt. $(SC, \leq)$.

The dynamic nature of web service composition results in the problem that typically a customer can neither know all subjects (i.e. web services) that will be involved in his request nor anticipate all types of information that will be communicated between the web services. As a consequence, his initially formulated security policy will cover only a fraction of the involved subjects and data and has to be extended accordingly on the fly by involved web services. We will sketch this *conservative* extension of security policies in the following.

Since a customer usually does not know about all available web services, the web service clearance σ_{ws} provided by the customer may be undefined for some unknown web services WS. Suppose that another web service WS' acting on behalf of the customer creates a plan involving the provision of confidential data to WS. Since the plan violates the security requirements (we assume that unknown services have no clearences at all), it would be rejected when checked by the type calculus presented in Section 4. However, we can incorporate a delegation mechanism that allows the customer to delegate a web service WS' the right to extend σ_{ws} to a new mapping σ'_{ws} that coincides with σ_{ws} in all its defined values (i.e. $\sigma_{ws}(x) = \sigma'_{ws}(x)$ for all $x \in domain(\sigma_{ws})$). Therefore, the customer provides a *delegation classification* that is again a partial mapping σ_{del} from the set of web services to SC. $\sigma_{del}(WS)$ denotes the maximal clearance a web service WS may allocate to an unknown web service. Let $\perp$ be the bottom element of SC then $\sigma_{del}(WS) = \perp$ denotes that WS is not allowed to classify any previously unknown web service. Let $\top$ be the top element in SC, then $\sigma_{del}(WS) = \top$ gives WS full discretionary power with respect to the classification of previously unknown web services. Notice however, that once the customer or some web service with appropriate delegation rights has fixed the clearance of a web service it cannot be changed anymore.

As mentioned before, we consider SC as a set of tuples $\tau_1, \dots, \tau_n$. Each tuple reflects the classification or clearance with respect to different security categories or aspects like, for instance, location information or payment details. Web services, like web interfaces to data bases, may act as data sources and formulate their own security requirements for the newly provided data. Besides classifying the new data according to the given categories they can introduce new security categories by introducing a new lattice $(SC', \leq')$ operating on tuples $\langle \tau_1, \dots, \tau_n, \tau_{n+1}, \dots, \tau_{n+m} \rangle$ such that for all tuples in SC': $\langle \tau_1, \dots, \tau_{n+m} \rangle \leq' \langle \tau'_1, \dots, \tau'_{n+m} \rangle$ implies $\langle \tau_1, \dots, \tau_n \rangle \leq \langle \tau'_1, \dots, \tau'_n \rangle$ and additionally, $\langle \tau_1, \dots, \tau_n \rangle \leq \langle \tau'_1, \dots, \tau'_n \rangle$ implies $\langle \tau_1, \dots, \tau_{n+m} \rangle \leq' \langle \tau'_1, \dots, \tau'_n, \tau_{n+1}, \dots, \tau_{n+m} \rangle$.

Analogously, the web service conservatively extends the mappings σ_{ws} and σ_{del} to σ'_{ws} and σ'_{del} which now operate on SC' instead of SC but are identical to the former mappings with respect to all previously existing categories, i.e. $\exists \tau_{n+1}, \dots, \tau_{n+m} : \sigma'_{ws}(WS) = \langle \tau_1, \dots, \tau_{n+m} \rangle$ iff $\sigma_{ws}(WS) = \langle \tau_1, \dots, \tau_n \rangle$ for all web services WS; and $\exists \tau_{n+1}, \dots, \tau_{n+m} : \sigma'_{del}(WS) = \langle \tau_1, \dots, \tau_{n+m} \rangle$ iff $\sigma_{del}(WS) = \langle \tau_1, \dots, \tau_n \rangle$ for all web services WS. All data provided by the call of such a web service are classified as $\perp$ with respect to newly introduced categories. Otherwise the web service could easily block information provided freely by the customer if it introduces a new category, classifies the provided data as high with respect to this category, and gives no clearance for this category to any web service. Similarly, we can *restrict* mappings σ_{ws} and σ_{del} by removing particular categories.

Example Revisited. Let us consider the example we presented in the introduction. The first step in this example is the specification of σ_{ws} by the customer. The customer selects two categories *location info* and *payment info* with H and L as potential values to formulate his security policy. Location info relates to informations about his whereabouts while payment info covers information about the details of his bank accounts or credit cards. He rates the web services as follows. Since he trusts in his secretary with respect to both categorizes, he chooses $\sigma_{ws}(SE) = \langle H, H \rangle$. However, he does not trust the travel agencies with respect to all his payment informations: $\sigma_{ws}(TA_i) = \langle H, L \rangle$ for $i \in \{1, 2\}$. Contrarily, he trusts both payment agents, paying for him the bills with respect to different accounts, in all his payment information. However, he only trusts the first agent in keeping also his whereabouts secret. The reason might be that the corresponding bank account belongs to some public agency and cash audits may reveal the details of the trip. Thus, we obtain $\sigma_{ws}(PA_1) = \langle H, H \rangle$ and $\sigma_{ws}(PA_2) = \langle L, H \rangle$.

4 Type Calculus To Enforce Security Policies

In the last section we described the mechanism to synthesize a common security policy for all participating web services that is consistent with the initially formulated policy of the customer. This section illustrates how a web service can prove whether the execution of its synthesized plan would violate the requested security policy and how it can classify newly computed data according to the given information flow policy.

Our approach is based on the work of D. Volpano and G. Smith [29, 25] on using type systems to secure information flow. Its underlying idea is as follows. We assume that a program has low-level and high-level inputs and computes some low-level and high-level outputs. Such a program is considered secure if the low-level output of the program does not depend on the high-level input. The basic idea is to monitor the data flow of high-level input with the help of a type system and prove that no high-level input was used to compute any low-level output.

Therefore, types are attached to all constructs in the programming language that store or provide information (like, for instance, locations, parameters or variables). Typing rules are formulated that specify the consequences to the involved types once any particular statement of the programming language is executed.

The programming language used in [29] is a simplified sequential programming language which provides conditionals, loops, and procedures. We extend this language by the additional feature of calling web services WS (in a blocking mode) by **call**(WS, x, y). Figure 1 presents the syntax of this language. Op and R are just place holders for the various functions and relations built into the language. Examples are $+$, $-$, $<$, and $=$. Metavariables x and P ranges over identifiers, l over locations and n over numerical literals.

The security classes SC of the underlying information flow policy $(SC, \leq)$ constitute the set of so-called data types which are used to classify data. Data are represented by expressions in the programming language. Hence, all expressions

$$\begin{array}{ll}
\text{(Expr)} & e ::= x \mid P \mid n \mid l \mid Op(e, e') \mid eRe' \mid \textbf{proc}(\textbf{in}\ x, \textbf{out}\ y)\ c \\
\text{(Comm)} & c ::= x := e' \mid c; c' \mid P(e, e') \mid \textbf{call}(WS, x, y) \mid \textbf{while}\ e\ \textbf{do}\ c \\
& \qquad \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' \mid \textbf{letvar}\ x := e\ \textbf{in}\ c \mid \\
& \qquad \textbf{letproc}\ P\ (\textbf{in}\ x, \textbf{out}\ y)\ c\ \textbf{in}\ c'
\end{array}$$

Fig. 1. Syntax of the language

Expr possess a data type τ denoting their classification. Intuitively, the type of *Expr* has to be equal to or higher than the least upper bound of the security types of all information we have used to evaluate *Expr*. For instance, given two expressions e and e' then $Op(e, e')$ obtains the least upper bound of the security types of e and e' as its security type.

Type calculus rules are used to propagate data types along expressions. The following rule illustrates this for expressions constructed with the help of a function *Op*. γ represents the assignment of the variables and is used to map occurring variables to their types.

$$\frac{\gamma \vdash e : \tau, \gamma \vdash e' : \tau}{\gamma \vdash Op(e, e') : \tau}\ Op$$

The rule makes use of an implicit type coercion: $\gamma \vdash e : \tau$ and $\tau \leq \tau'$ implies $\gamma \vdash e : \tau'$ ("upgrading data"). Hence we can always upgrade implicitly the type of e and e' to its least upper bound τ in order to apply the rule.

Variables are used to store information. They need a sort of clearance to hold classified information. This is denoted by a phrase type τ *acc*. $\gamma \vdash x : \tau\ acc$ says that x has the clearance to house data with a classification up to τ. Again there is a contravariant type coercion by $\gamma \vdash e : \tau\ acc$ and $\tau' \leq \tau$ implies $\gamma \vdash e : \tau' acc$ to simplify the following typing rule for assignments.

$$\frac{\gamma \vdash x : \tau\ acc, \gamma \vdash e : \tau}{\gamma \vdash x := e : \tau\ cmd}\ Assign$$

A phrase type τ *cmd* is assigned to a program fragment, like a statement or a block of statements, to store information about the security types of changed variables inside the fragment. If a fragment has type τ *cmd* then only variables with clearances τ and higher will be changed inside the block. The idea is that the execution of the fragment does not assign to variables of type τ' *var* for τ' lower than τ. It is thus permissible for the fragment to be executed conditionally depending on the values of variables labelled τ or lower without leaking information from the context into lower graded variables.

Variables play two roles: first, they provide data and have a classification τ and second, they store information and need a clearance τ *acc*. Both notions are integrated into a phrase type τ *var* combining the classification τ *cmd* and the clearance τ *acc*.

[29] provide also types and typing rules for procedures which we simplified for our purposes (since we omit in-out parameters). A phrase type $\tau\ proc(\tau_1, \tau_2\ acc)$ indicates that the call of the procedure has type τ *cmd*, the input parameter has the classification τ_1 and the output must have the clearance τ_2 *acc*. $[e' \backslash P]e$ denotes the capture-avoiding substitution of e' for all free occurrences of P in e. Then the rules for declaring (polymorphic) procedures are as follows.

$$\frac{\begin{array}{l}\gamma \vdash \mathbf{proc}(\mathbf{in}\ x_1, \mathbf{out}\ x_2)c : \pi, \\ \gamma \vdash [\mathbf{proc}(\mathbf{in}\ x_1, \mathbf{out}\ x_2)c\ /\ P]c' :\ \tau\ cmd\end{array}}{\gamma \vdash \mathbf{letproc}\ P(\mathbf{in}\ x_1, \mathbf{out}\ x_2)\ c\ in\ c' : \tau\ cmd}\ LetProc$$

$$\frac{\gamma[x_1 : \tau_1,\ x_2 : \tau_2\ acc] \vdash c : \tau\ cmd}{\gamma \vdash \mathbf{proc}(\mathbf{in}\ x_1, \mathbf{out}\ x_2)c :\ \tau\ proc(\tau_1, \tau_2\ acc)}\ Proc$$

The rule for typing the application of procedures is straightforward:

$$\frac{\begin{array}{l}\gamma \vdash P :\ \tau\ proc(\tau_1, \tau_2\ acc),\\ \gamma \vdash e_1 : \tau_1,\\ \gamma \vdash e_2 : \tau_2\ acc\end{array}}{\gamma \vdash P(e_1, e_2) : \tau\ cmd}\ ApplyProc$$

The idea of our approach is that web service calls can be treated like procedures. We consider web services as external procedures. We can encode global states as global variables common to various web services. In contrast to procedures, web services have to be first class citizens in our approach possessing their individual clearances. The call of a web service has to be guarded by a check whether the input to be provided to the service lies within its clearance. Thus we introduce a new phrase type $\tau\ proc(\tau_1, \tau_2)\ \tau'$ to type web services. First, the phrase denotes that the call of the web service has type τ *cmd*. This information is only required in the presence of common variables (global states) of calling and called web services. Second, the input has the classification τ_1 and the resulting output requires the clearing τ_2. Additionally, τ' defines the clearance of the web service from the caller's point of view. It is crucial to understand the difference between the classification τ_1 of the input parameter and the clearance τ' which the calling web service assigns to the called web service. In general, web services are polymorphic in their types, for example, τ_1 and τ_2 are type variables that are related by the constraint $\tau_1 = \tau_2$. In this case the output would require the same clearance as the classification of the input. An example would be a web service that simply copies its input to the output. Thus τ_1 and τ_2 reflect the constraints on the security types considering the computation inside the called web service while τ' reflects the trust the calling web service has in the called web service.

The rule for typing the call to the web services is similar to the corresponding rule for procedure calls:

$$\frac{\begin{array}{l}\gamma \vdash WS :\ \tau\ proc(\tau_1, \tau_2\ acc)\tau',\\ \gamma \vdash e_1 : \tau_1,\\ \gamma \vdash e_2 : \tau_2\ acc\\ \gamma \vdash \tau_1 \leq \tau'\\ \gamma \vdash \tau_2 \leq \tau'\end{array}}{\gamma \vdash \mathbf{call}(WS, e_1, e_2) : \tau\ cmd}\ ApplyWS$$

There is, however, one difference between a sub web service call and a procedure call: while we know the body of a procedure being part of the program itself, we do not know the interior of called web services because firstly, there is no fixed program so no fixed body can be exported as part of the specification of a web service, and secondly because in general owners of web services do not want to disclose the source code of their web services. Instead a web service will export its security type (considered as a procedure) $\tau\ proc(\tau_1, \tau_2\ acc)$ as part of its overall specification. Given the inherited security policy σ_{ws} of the customer and the published security type $\tau\ proc(\tau_1, \tau_2\ acc)$ of a suitable web service WS' to be called by a web service WS, WS can assemble the type of WS' to $\tau\ proc(\tau_1, \tau_2\ acc)\ \sigma_{ws}(WS')$. Contrarily, the called web service requires the type of each parameter. Hence, the calling web service provides the security type of each parameter as intrinsic part of a call.

Example Revisited. Let us consider the example we presented in the introduction. In Section 3 we illustrated already how he customer chooses his security policy. In a next step the customer instructs his secretary agent SE to organize his trip to Berlin and Rome. The trip to Berlin is not classified at all, i.e. $\langle L, L\rangle$ while the second trip contains confidential location information: $\langle H, L\rangle$. The secretary agent constructs a plan of how to decompose the task into a sequence of web service calls and comes up with the following program:

$$\begin{array}{l}\mathbf{call}(TA_1, \mathit{flight}_{Berlin}, \mathit{price}_{Berlin});\\ \mathbf{call}(TA_2, \mathit{flight}_{Rome}, \mathit{price}_{Rome});\\ \mathbf{call}(PA_1, \mathit{price}_{Berlin}, \mathit{okP});\\ \mathbf{call}(PA_1, \mathit{price}_{Rome}, \mathit{okP});\end{array}$$

Both travel agents publish $H\ proc(X, X\ acc)$ (X being a type variable) as their specification of their security types. This means that the output (the price of the flight) requires the same security class as the input (the flight requirements). It reflects the fact that the price is calculated with the help of the flight information. Suppose, a travel agent would offer a flat rate for European flights, i.e. each flight in Europe would cost the same amount of money, then he could publish a security type $H\ proc(X, L\ acc)$. Both payment agents specify $H\ proc(X, L\ acc)$ as their security types since we assume that their acknowledgments okP do not depend on the prices.

Based on these typings, the type calculus rates price_{Berlin} as L wrt. location information and price_{Rome} as H. As a consequence, the secretary agent cannot use PA_2 to pay the flight to Rome because it only has a L-clearance with respect to location information. If we suppose that the secretary makes use of a flat rate-offer then price_{Rome} would be rated as L which would enable the use of PA_2 to pay the flight.

5 Related Work

The dynamic composition of web services is a quite new research area coming up within the last four years. [21] gives a survey about the different AI-planning approaches to tackle dynamic composition. Since our approach is independent of the way a plan is generated we will not go into the details of these approaches.

With the advent of pervasive computing, a lot of difficulties arouse with respect to the transmitted, stored, used and computed sensitive data. There are various approaches concerned with trust management (e.g. [12, 13]) and authentication techniques, i.e. how to ensure that a web service does not try to cheat [22] because cheating would not gain any benefits in the underlying economic model.

Various aspects of security policy of web services have been investigated. Some aspects were concerned with how to specify a policy in a machine readable and user friendly way at the same time (see e.g. IBM and Micosoft's Web Services security specification [11], especially the WS-Policy part), how to compose different policies and how to prove that the web service does ensure its policy specification with each request. Current approaches concentrate on access control. So the plan is executed in any case and if the access control matrix forbids any access during the execution, it stops and a new plan has to be created. The approaches can be distinguished with respect to the type of policy they work with: e.g. KAoS [27] and Ponder [8], which handle security policies for authentication and obligations, or REI [20], which works with security policies for rights, prohibitions, obligations and dispensations. There are also two approaches which concentrate on the composition of security policies independent of the type of the policy: The main idea, Samarati et al. present in [5] and a practical extension is introduces with IBM's algebra for composing policies based on their Enterprise Privacy Authentication Language (EPAL) [11], [26].

Starting with the work of Goguen and Meseguer, information flow control has been subject of a large variety of different approaches introducing different formal notions of independence. Most prominent, McLean [18], Zakinthinos and Lee [32] and Mantel [16] proposed frameworks to embed these different notions in a uniform framework. Our work is based on language-based information flow. The general problem whether a program leaks information from high-level to low-level is undecidable. Thus, type calculi as they are proposed, for instance, in [29] are incomplete. Meanwhile following Volpano and Smiths work, more refined type calculi (e.g. [23]) have been developed that are able to recognize more programs as secure. They are, for instance, able to detect if different program paths will result in the same low-level results. In this case the condition whether to take the one path or the other can freely use high-level inputs. However, the power of these approaches result in a more expensive computation and propagation of types. Since dynamically composed web services are rather simple programs, we decided to use a less refined type calculus, which require less resources.

6 Conclusion

We presented an approach to use language-based information flow control to ensure the confidentiality (as well as integrity) of user's data provided to dynamically composed web services. The data flow of confidential data is monitored by a type calculus which propagates the security requirements of the user along the planning and execution of dynamically composed web services.

Future work concern the development of an appropriate language to encode the security requirements σ_{ws} of a customer. Possible solutions are based on the use of appropriate description logics [1]. Moreover, we will analyze the application of negotiation mechanisms in case the web service does not fit with the policy of a web service which should be called. Here the SLAng approach [15] might be helpful.

Another problem in this context arises if the web service executes its plan but one of the chosen web services is not available anymore. An easy solution would be to try to find another web service with the same policy and functional specification. But if the web service cannot find such a web service, we do not want to start from scratch again by creating a new plan for the whole task, but we would like to patch the existing plan by creating only a plan for the changed sub tasks. In order to come up with such a solution we have to investigate how the change of the security types of some variables or parameters will effect existing proofs of the type calculus.

Acknowledgements

We are grateful for many fruitful and inspiring discussions that we had with our former colleague Axel Schairer on this work in general and on earlier versions of this paper in particular. We would also like to thank the anonymous referees for their helpful feedback on this work.

References

1. Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Mechanizing Mathematical Reasoning, Festschrift in Honor of J.H.Siekmann*. Springer-Verlag, LNCS 2605, 2005.
2. D. E. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report MTR-2997, MITRE, 1976.
3. V. Richard Benjamins, Enric Plaza, Enrico Motta, Dieter Fensel, Rudi Studer, Bob Wielinga, Guus Schreiber, and Zdenek Zdrahal. Ibrow3 - an intelligent brokering service for knowledge-component reuse on the world wide web. In *11th Knowledge Acquisition for Knowledge-Based System Workshop (KAW98)*, 1998.
4. T. Berners-Lee, J. Hendler, J., and O. Lassila. The semantic web. *Scientific American*, May 2001.
5. P.A. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, 2002.
6. J. Bryson, D. Martin, S.I. McIlraith, and L.A. Stein. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web. In Ning Zhong, Jiming Liu, and Yiyu Lao, editors, *Web Intelligence*. Springer Verlag, 2002.
7. DAML-S DARPA agent markup language for services, version 0.9. `http://www.daml.org/services/daml-s/0.9/daml-s.html`.
8. Naranker Dulay, Nicodemos Damianou, Emil Lupu, and Morris Sloman. A policy language for the management of distributed agents. In *Agent Oriented Software Engineering, AOSE*, pages 84–100. Springer, 2001.
9. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20, Oakland, CA, USA, 1982.

10. J. A. Goguen and J. Meseguer. Inference Control and Unwinding. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 75–86, Oakland, CA, USA, 1984.
11. IBM and Microsoft. *Security in a Web Service World: A proposed architecture and roadmap.* www-106.ibm.com/developerworks/webservices/library/ws-secmap, April 2002.
12. L. Kagal, T. Finin, and A. Joshi. Trust based security for pervasive computing enviroments. *IEEE Computer*, 24(12):154–157, December 2001.
13. L. Kagal, T. Finin, and A. Joshi. Developing secure agent systems using delegation based trust management. In K. Fischer and D. Hutter, editors, *Security of Mobile MultiAgent Systems (SEMAS 02) held at Autonomous Agents and MultiAgent Systems (AAMAS 02)*, 2002.
14. M. Klusch, A. Gerber, and M. Schmidt. Semantic web service composition planning with owls-xplan. 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, 2005.
15. D. Davide Lamanna, James Skene, and Wolfgang Emmerich. Slang: A language for defining service level agreements. In *IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS*, pages 100–, 2003.
16. H. Mantel. Possibilistic Definitions of Security – An Assembly Kit. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, UK, 2000.
17. J. D. McLean. Proving Noninterference and Functional Correctness using Traces. *Journal of Computer Security*, 1(1):37–57, 1992.
18. J.D. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of IEEE Symposium on Security and Privacy.* IEEE Computer Society, 1994.
19. OWL ontology web language, w3c standard technical recommendation. `http://www.w3.org/TR/2003/WD-owl-ref-20030331/`.
20. Anand Patwardhan, Vlad Korolev, Lalana Kagal, and Anupam Joshi. Enforcing policies in pervasive environments. In *MobiQuitous*, pages 299–308, 2004.
21. Joachim Peer. Web service composition as ai planning - a survey. Technical report, University of St. Gallen, March 2005.
22. S.D. Ramchurn, D. Huynh, and N.R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.
23. A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), 2003.
24. M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. In *Proceedings of AAMAS 2003 Workshop on Web Services and Agent-Based Engineering*, 2003.
25. Geoffrey Smith and Dennis Volpano. Secure information flow in a multi-threaded imperative language. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 355–364, New York, NY, 1998.
26. William H. Stufflebeam, Annie I. Antón, Qingfeng He, and Neha Jain. Specifying privacy policies with p3p and epal: lessons learned. In *Workshop on Privacy in the Electronic Society, WPES*, page 35, 2004.
27. Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Austin Tate, and Jeff Dalton. Applying kaos services to ensure policy compliance for semantic web services workflow composition and enactment. In *International Semantic Web Conference*, pages 425–440, 2004.

28. Dennis M. Volpano and Geoffrey Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.
29. Dennis M. Volpano and Geoffrey Smith. A type-based approach to program security. In *TAPSOFT*, pages 607–621, 1997.
30. R. Waldinger. Deductive composition of web software agents. In *NASA Workshop on Formal Approaches to Agent-Based Systems*. Springer-Verlag, LNCS 1871, 2000.
31. D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, pages 20–23, Sanibel Island, Florida, USA, October 2003.
32. A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 94–102, Oakland, CA, USA, 1997.

Analysing a Biometric Authentication Protocol for 3G Mobile Systems Using CSP and Rank Functions

Siraj A. Shaikh[1] and Christos K. Dimitriadis[2]

[1] Department of Computing, UGBS, University of Gloucestershire, Park Campus, Cheltenham Spa, GL52 2RH, UK
sshaikh@glos.ac.uk
[2] University of Piraeus, 80 A. Dimitriou, 18534 Piraeus, Greece
cricodc@unipi.gr

Abstract. We study a protocol, called BIO3G, which provides biometric-based user authentication and key establishment in Third Generation (3G) mobile environments. BIO3G provides end-to-end user authentication to the mobile operator, requiring no storing or transferring of biometric data and, eliminating the need for biometric enrolment and administration, which is time-consuming for the user and expensive for the mobile operator. We model BIO3G using process algebra Communicating Sequential Processes (CSP) and verify it using Schneider's rank functions.

1 Introduction

3G mobile systems offer true broadband data transmission, opening the path for the provision of new and improved services. The two dominant standards are the Universal Mobile Telecommunications System (UMTS), developed by the 3rd Generation Partnership Project (3GPP) which is a joint initiative of telecommunication standardization organizations from the US, Europe, Japan and Korea, and, the Code-Division Multiple Access 2000 (CDMA2000), developed by a separate partnership of standardisation organisations called 3GPP2. Both are designed to support a wide range of multimedia services, with enhanced performance, security and cost effectiveness.

User authentication is a primary element of the 3G network access security mechanism, which is usually implemented by the use of a Personal Identification Number (PIN) [1]. The rest of the process relies on the authentication of pre-stored secrets, such as cryptographic keys, or identifiers such as the International Mobile Subscriber Identity (IMSI) but not the user. Furthermore, knowledge as well as the possession of an item, does not distinguish a person uniquely, revealing an inherent security weakness of password and token-based authentication mechanisms. Moreover, PIN stealing, guessing or cracking have become very popular, with software tools implementing relevant attacks and research papers describing sophisticated techniques for invading PIN security [2]. Modern biometric technologies provide enhanced security levels by introducing a new dimension in the authentication process called "proof by property". Biometrics is defined *as the automatic use of human physiological or behavioural characteristics to determine or verify an identity* [3]. The design and deployment of a security architecture incorporating biometric technologies, however, hides many pitfalls that if underestimated can lead to major security weaknesses and threats [4].

In this paper, we study a protocol, called BIO3G, which provides biometric-based user authentication and key establishment in Third Generation (3G) mobile environments. The protocol, introduced in [5], extends the UMTS Authentication and Key Agreement (UMTS-AKA) mechanism [6], which is part of the 3GPP standard, to incorporate the use of biometrics to provide user authentication and key establishment. The protocol provides an enhanced alternative to the common practice of utilising biometrics locally for gaining access to the device to. It provides end-to-end user authentication to the mobile operator, requiring no storing or transferring of biometric data and, eliminating the need for biometric enrolment and administration, which is time-consuming for the user and expensive for the mobile operator. BIO3G substitutes the weak PIN mechanism upon which network access security relies and proposes an alternative to local biometric authentication, which is commonly deployed for gaining access to the mobile device. We model BIO3G using CSP [7] and specify its authentication properties as trace specifications. We then verify the protocol using Schneider's rank functions. The rest of this paper is organised as follows. Section 2 presents the basics of 3G Network access security. Section 3 describes the BIO3G protocol in informal terms. Section 4 introduces Schneider's rank functions approach to verifying protocols. Section 5 presents the CSP and rank functions analysis of the BIO3G protocol. We conclude the paper with a discussion in Section 6.

2 UMTS Network Access Security

BIO3G utilises the existing network access authentication mechanism of UMTS, as specified by the 3GPP standards for 3G security [6]. These standards describe the UMTS-AKA mechanism as their core element for entity authentication, user identity management, confidentiality and integrity. The UMTS-AKA mechanism is based on a 128-bit secret key (*K*), which is pre-shared between the mobile operator and the USIM. The USIM is a cryptography-enabled smart card identified by a 15 digit number called IMSI. The USIM authenticates the user, by the use of a PIN. Figure 1 presents a summary of the elements of UMTS-AKA.

Mutual authentication between the USIM and the mobile operator is realized by a challenge and response mechanism. A random number (*RAND*) is calculated by the mobile operator and submitted to the USIM, along with a value (*AUTN*) derived by the combination of *RAND* and *K* with a number of parameters, including a sequence number. The USIM authenticates the mobile operator by analysing and verifying *AUTN*. The USIM computes a value (RES), by applying *RAND* and *K* to a function (f2), and submits it to the mobile operator for verification (comparison with similarly computed *XRES*), realizing the authentication of the USIM. Figure 1 presents a summary of the UMTS-AKA mechanism.

The USIM uses the UMTS ciphering algorithm (f8) to provide confidentiality, which produces a KeyStream Block (*KSB*) using a 128-bit Cipher Key (*CK*) and a number of other parameters. Integrity protection is implemented by the deployment of a 128-bit Integrity Key (*IK*), which is used for the calculation of Message Authentication Codes (*MAC*). The *CK* and *IK* are computed by the USIM and the mobile operator, by applying the pre-shared *K* and *RAND* to key generation functions (f3 and f4

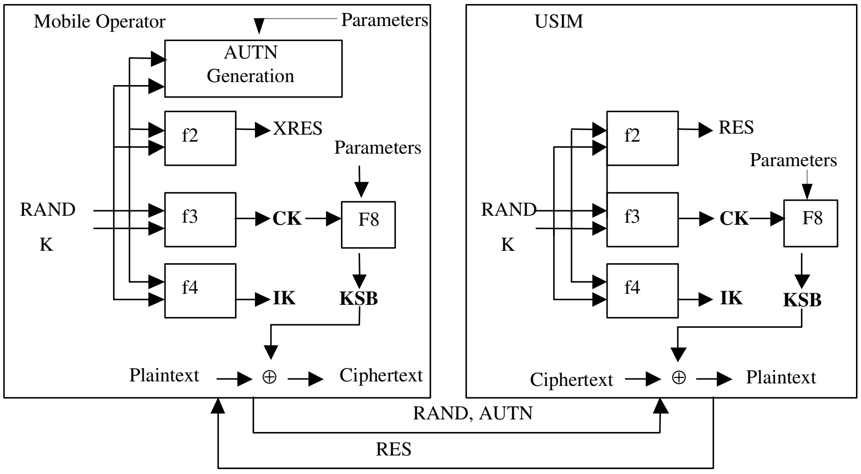


Fig. 1. UMTS-AKA Network access security

respectively). The *CK*, *IK*, *AUTN*, *RAND* and *XRES* compose a group of UMTS-AKA authentication elements, called *Authentication Vector* (*AV*).

To summarise, the user is authenticated only locally in the USIM by the provision of a PIN. The USIM utilises two pre-stored values, IMSI for identifying and a key *K* for authenticating the user to the mobile operator, resulting in the whole security infrastructure to depend on a simple PIN mechanism.

3 The BIO3G Protocol

We identify three entities that participate in the protocol and denote the User as *U*, the UMTS Subscriber Identity Module as *USIM* and the Mobile Operator as *MO*. We represent a non-invertible value *B* derived from a biometric sample *A*, provided by *U*, as $B = f_{fe}(A)$, where f_{fe} is a randomness generating function [8]. For the purpose of our specification, we model the function f_{fe} to have two important properties:

- f_{fe} is one-to-one and error-tolerant (for a specific maximum space of differences *d*) [8], that is, for some $A' = A + T$, where T symbolizes the differences – depending on the metric – between *A* and *A'*, if $T > d$ then $f_{fe}(A) \neq f_{fe}(A')$, else if $T < d$ then $f_{fe}(A) = f_{fe}(A')$.
- f_{fe} is a one-way function, that is, given *B* it is computationally hard to find *A* such that $f_{fe}(A) = B$.

The BIO3G protocol relies on UMTS-AKA for some pre-shard values between the entities, which form the core message components in the protocol. We assume both *USIM* and *MO* are in secure possession of *K*, *CK*, *IK* and *KSB* and, are aware of each other with respect to these.

We divide the protocol into two phases. The first phase is concerned with obtaining a biometric sample *A* from a user *U* and deriving a non-invertible value *B* using the randomness generating function described above, such that $B = f_{fe}(A)$. The User Equipment (*UE*) is responsible for obtaining *A* from *U* and calculating *B*. The *UE* then passes the value of *B* onto *USIM*. We assume the communication between *UE* and *USIM* is internal and therefore reliable as per the specifications of 3GPP [9].

The second phase involves a single step of transmission from *USIM* to *MO*, allowing *MO* to authenticate *U*. This is achieved by deriving a new symmetric key *K′* using *B*, which will replace the original key *K* for subsequent UMTS-AKA procedures. Note that the exchange of values between the *UE* and *MO*, for the establishment of the new *K′*, takes place only during the initial run of the BIO3G protocol. When the user *U* reconnects to the network, a new key will be established between *USIM* and *MO* using a new biometric measurement and the existing key *K*. If the value of the new key is correct, UMTS-AKA will succeed and positively authenticate the user to the *MO*. In case of a mismatched biometric sample, that is a different value of *B*, UMTS-AKA will fail to authenticate the user. We specify the phase 2 of the protocol informally in Figure 2.

$$(1) \qquad USIM \rightarrow MO \qquad : \qquad (KSB \oplus D, |KSB \oplus D|_{IK})$$

Fig. 2. BIO3G protocol

where the value *D* is an offset of *B* calculated against the exclusive-or’d combination of *CK* and *IK*, such that $D = (CK \oplus IK) - B$, and the actual message of the protocol is *KSB* exclusive-or’d with the value of *D*, $KSB \oplus D$, concatenated with the MAC-I (Message Authentication Code for Integrity) value of $KSB \oplus D$ using the key *IK*. The purpose of MAC-I is to provide data integrity for the encrypted message ($KSB \oplus D$) in the protocol. It is calculated using an integrity algorithm, known as f_9 from the UMTS-AKA standard. In more detail, for some message *M*, we denote MAC-I produced by the f_9 algorithm using a key *IK* as $|M|_{IK}$, along with the following parameters

$$|M|_{IK} = f_9(IK, COUNT\text{-}I, FRESH, DIRECTION, M)$$

where *IK* is a pre-shared *Integrity Key*, *COUNT-I* is a sequence number used for the integrity of the communication between the entities, *FRESH* is a nonce used to avoid message replay and *DIRECTION* is used to distinguish between the direction of the message. Once the actual message is received by *MO*, it performs two main operations:

1. checks the integrity of the message $KSB \oplus D$, by computing the MAC-I value for it and comparing it with the MAC-I value sent. If the values do match, then this validates data integrity for the sent message, and
2. derives a new shared key *K′* - by first recovering *D* from the message such that, $D = KSB \oplus (KSB \oplus D)$, and then computing *B* as an offset of *D* against the exclusive-or’d combination of *CK* and *IK* such that, $B = (CK \oplus IK) - D$, and finally, deriving the key *K′* such that, $K' = f_3(K,B)$ (function f_3 is part of UMTS-AKA standard). Implicit here is also *MO*’s authentication of *U*, by way of *U*’s sample *A* – since *B* is derived as $B = f_{fe}(A)$ and *K′* is derived as $K' = f_3(K,B)$, the establishment of *K′* by *MO* allows it to authenticate *U*.

4 Schneider's Rank Functions Approach

In this section, we introduce Schneider's rank functions approach to verifying protocols. We take for granted the reader's basic knowledge of CSP [10,11,12].

4.1 Schneider's Model of the Network

Schneider [13] models the protocol as a network where the participants are modelled as CSP processes acting in parallel. An intruder process is also modelled with capabilities as defined by Dolev and Yao [14], including blocking, replaying, spoofing and manipulating any messages that appear on any of the public channels in the network. To express message transmission and reception for each process, Schneider introduces two channels, *send* and *receive*, which are public channels that all processes use to send and receive messages by. The events are structured as *send.i.j.m* where a message *m* is sent by source *i* to destination *j* on the channel *send* while *receive.j.i.m* represents a message *m* being received by *j* from a source *i* on the channel *receive*. We consider a set of users $\mathcal{U}$ to represent all the participants that use the network and *Intruder* to denote the intruder process. For each participant $i \in \mathcal{U}$, a CSP process $USER_i$ represents the behaviour of the participant. We specify the complete network **NET** as **NET** = $(|||_{i\in\mathcal{U}} USER_i) \underset{(send,\, receive)}{||} Intruder$ where all participants in $\mathcal{U}$ are forced to synchronise with *Intruder* on *send* and *receive* channels. In order to model the capabilities of the intruder according to the Dolev and Yao [14] model, Schneider [13] introduces a generates '$\vdash$' relation to characterise what messages may be generated from a given set of messages. The rules that define this relation are as follows, where *S* is some set of messages, *m* is a message and *k* is some key

- $m \in S$ then $S \vdash m$
- $S \vdash \{m\}_k \wedge S \vdash k \Leftrightarrow S \vdash m$
- $S \vdash m$ and $S \subseteq S'$ then $S' \vdash m$
- $S \vdash m_1 .m_2 \Leftrightarrow S \vdash m_1 \wedge S \vdash m_2$
- $S \vdash m \wedge S \vdash k \Rightarrow S \vdash \{m\}_k$
- $S \vdash m_1 \wedge S \vdash m_2 \Leftrightarrow S \vdash m_1 .m_2$
- $S \vdash m_i$ for each $m_i \in S'$ and $S' \vdash m$ then $S \vdash m$

We use this relation to specify a recursive definition of *Intruder* as follows:

$$Intruder(S) = send.i.j.m \rightarrow Intruder(S \cup \{m\})$$
$$\Box$$
$$\underset{i,j\in\mathcal{U},S\vdash m}{\Box} receive.i.j.m \rightarrow Intruder(S)$$

The *Intruder* process is parameterised by a set of messages *S* that denotes the set of messages in the possession of the intruder. Such a set of initial knowledge is denoted as *Initial Knowledge*, ***IK***, and the *Intruder* is specified such that *Intruder*(***IK***).

4.2 Rank Functions

Let the set of participants on the network be $\mathcal{U}$, the set of nonces used by the participants in protocol runs as $\mathcal{N}$ and a set of encryption keys used as $\mathcal{K}$. The set of all atoms is $\mathcal{A}$, where $\mathcal{A} = \mathcal{U} \cup \mathcal{N} \cup \mathcal{K}$. Let the message space $\mathcal{M}$ contain all the messages and signals that may appear during a protocol's execution, such that $m \in \mathcal{A} \Rightarrow m \in \mathcal{M}$. Schneider [13] defines a rank function ρ to map events and messages to integers $\rho\colon \mathcal{M} \rightarrow \mathbb{Z}$. The message space is then divided into two parts where $\mathcal{M}_{\rho-} = \{m \in \mathcal{M} \mid \rho(m) \leqslant 0\}$ and $\mathcal{M}_{\rho+} = \{m \in \mathcal{M} \mid \rho(m) > 0\}$.

The purpose of this partition of the message space is to characterise those messages that the intruder might get hold of without compromising the protocol – assigned a positive rank – and those messages that the enemy should never get hold of – assigned a non-positive rank. It is desirable for a process never to transmit a message of non-positive rank. For a certain process P to maintain positive rank, it is understood that it will never transmit a message with a non-positive rank unless it has previously received a message with a non-positive rank. More formally, for a process P,

$$P \textbf{ maintains } \rho \Leftrightarrow \forall\, tr \in traces(P) \bullet \rho(tr \Downarrow receive) > 0 \Rightarrow \rho(tr \Downarrow send) > 0$$

In other words P will never transmit any message m of $\rho(m) \leqslant 0$ unless it has received some m' of $\rho(m') \leqslant 0$ previously, with respect to some rank function ρ. It is not important who the message is received from or is sent to. Schneider [13] presents a general-purpose rank function theorem that ensures the messages that an *Intruder* gets hold of do not compromise the security property that the protocol provides.

RANK FUNCTION THEOREM

If, for sets R and T, there is a rank function $\rho\colon \mathcal{M} \rightarrow \mathbb{Z}$ satisfying

R1) $\forall\, m \in \boldsymbol{IK} \bullet \rho(m) > 0$

R2) $\forall\, \boldsymbol{S} \subseteq \mathcal{M}, m \in \mathcal{M} \bullet ((\forall m' \in \boldsymbol{S} \bullet \rho(m') > 0) \wedge \boldsymbol{S} \vdash m) \Rightarrow \rho(m) > 0$

R3) $\forall\, t \in T \bullet \rho(t) \leqslant 0$

R4) $\forall\, i \in \mathcal{U} \bullet User_i \underset{R}{\|} Stop \textbf{ maintains } \rho$

then **NET sat** R **precedes** T

The theorem states that if the rank function, and therefore the underlying **NET**, satisfies the four properties, then no messages of non-positive rank can circulate in $\textbf{NET} \underset{R}{\|} Stop$. In particular, an intruder should not be able to generate any illegal messages from the messages it knows at the beginning of the protocol from the set ***IK***, nor from the messages it sees during the protocol execution, denoted by a set $\boldsymbol{S}$. Also, honest participants should not be able to generate any illegal messages unless they are sent one, that is, every honest process maintains ρ while being restricted on R. The actual verification of the theorem conditions is performed manually for every rank function constructed for a protocol. Verifying different specifications may require different rank functions to be constructed for the same protocol.

5 Analysing BIO3G Using CSP and Rank Functions

In this section, we model BIO3G in CSP. We specify the different processes involved and, formalise the authentication properties of the protocol as trace specifications in Section 5.1. In Section 5.2, we describe our proof strategy and construct a rank function for BIO3G in Section 5.3 and verify the protocol in Section 5.4.

We specify the protocol in CSP and model different processes to represent the different entities taking part in the protocol. We model different types of channels for CSP events to represent the different types of communications between the entities. We use a *biosample* channel to abstract away the obtaining of a biometric sample from a biometric reader, which we assume is part of the user equipment. We use *submit* and *accept* channels, which we consider to be private between *UE* and *USIM* and, models the internal communication within *UE*. We use *send* and *receive* channels to represent the 3G network, which is considered to be public and hostile. We define three processes *UE*, *USIM* and *MO* and specify them as follows.

$$\begin{aligned} UE \;=\;& biosample?a \rightarrow \\ & Running.UE.USIM.a \rightarrow \\ & submit.UE.USIM!f_{fe}(a) \rightarrow Stop \end{aligned}$$

$$\begin{aligned} USIM(ksb,ck,ik,k) \;=\;& accept.USIM.UE?b \rightarrow \\ & Running.USIM.mo.b.k' \rightarrow \\ & send.USIM!mo!(ksb\oplus((ck\oplus ik)-b),|ksb\oplus((ck\oplus ik)-b)|_{ik}) \rightarrow Stop \end{aligned}$$

$$\begin{aligned} MO(ksb,ck,ik,k) \;=\;& receive.MO?usim?(ksb\oplus d,|ksb\oplus d|_{ik}) \rightarrow \\ & Commit.MO.usim.(ck\oplus ik)-d).(f_3(k,((ck\oplus ik)-d))) \rightarrow Stop \end{aligned}$$

Observe that we parameterise the CSP processes such that the values of a keystream block *ksb*, the cipher key *ck*, the Integrity Key *ik* and a pre-shared key *k* are all passed onto *USIM* and *MO* as perfect values from the underlying UMTS-AKA mechanism. The user equipment process *UE* is modelled to represent obtaining a biometric sample from a user, computing a non-invertible value using the f_{fe} function and passing it onto *USIM*. The *USIM* process accepts the f_{fe} computed value and essentially executes phase 2 of the protocol. We use these processes to model the entire network in CSP. We model a network **NET** as

$$\mathbf{NET} = ((UE_{\{submit\}}||_{\{accept\}}USIM(ksb,ck,ik,k)) \;|||\; MO(ksb,ck,ik,k)) \;||\; Medium$$

where $(UE_{\{submit\}}||_{\{accept\}}USIM(ksb,ck,ik,k))$ represents a combined entity in which *UE* and *USIM* operate. This entire system runs in parallel with the mobile operator *MO(ksb,ck,ik,k)*. We combine these two entities in an interleaving parallel composition. We then model the *Medium* process to represent the actual 3G network through which both entities communicate with each other. The *Medium* process is replaced by a particular *Intruder* process for this protocol in Section 5.2.

5.1 Specifying Authentication Properties

We illustrate a specific run of the BIO3G protocol, in Figure 3, using signal events to indicate the various stages of the protocol relevant to us. The *Running.UE.USIM.A* signal indicates *UE*'s submission of the biometric sample *A* to *USIM*. Although this sample is submitted in the form of a non-invertible value *B*, this signal is important as it indicates the submission a biometric sample on behalf of some user *U*. The *Running.USIM.MO.B.K'* signal indicates *USIM(KSB,CK,IK,K)*'s intention to run the protocol with *MO(KSB,CK,IK,K)* using the non-invertible value *B* (to identify this unique run of the protocol) and the new key *K'* to signify the establishment of this key. The *Commit.MO.USIM.B.K'* signal indicates *MO(KSB,CK,IK,K)'s* authentication of *USIM(KSB,CK,IK,K)*'s involvement in this run, where *B* indicates this unique run of the protocol and *K'* indicates the key established as a result of this run.

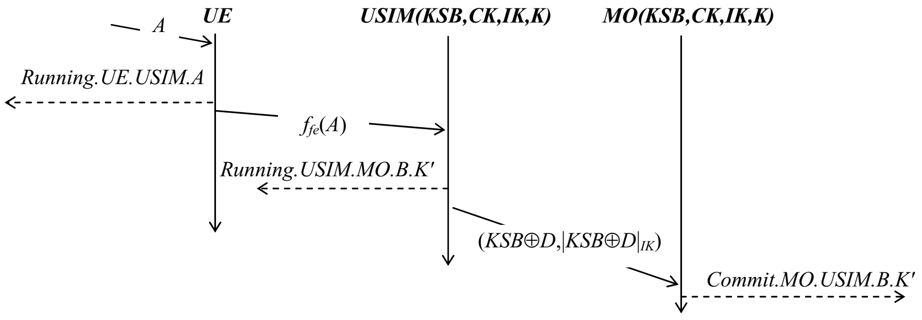


Fig. 3. An illustration of a BIO3G protocol run

We use *Running* and *Commit* signal events to explicitly specify two authentication properties for this protocol. In *Definition 1*, we specify an entity authentication property where the mobile operator *MO* authenticates (and establishes a key with) the *USIM* module using the non-invertible value *B*. This then allows us to specify the main goal of the protocol in *Definition 2*, that is the mobile operator's authentication of the biometric sample processed by the user equipment, and hence the user.

Definition 1. *BIO3G_Auth_Key*(*tr*) =

$$tr' \wedge \langle Commit.MO.USIM.B.K'\rangle \leqslant tr \Rightarrow \langle Running.USIM.MO.B.K'\rangle \text{ in } tr'$$

$$\wedge \ \#(tr \upharpoonright Running.USIM.MO.B.K') \geqslant \#(tr \upharpoonright Commit.MO.USIM.B.K')$$

The first clause in the *BIO3G_Auth_Key* specification specifies the causal precedence of *Running.USIM.MO.B.K'* over *Commit.MO.USIM.B.K'*. This means that every time a *Commit.MO.USIM.B.K'* event occurs, it is preceded by a *Running.USIM.MO.B.K'* event. The second clause specifies an injective agreement between the two runs, that is, for every *Commit.MO.USIM.B.K'* there is a unique *Running.USIM.MO.B.K'* that precedes it. The second clause is important because the protocol provides key establishment between the two entities; the use of the unique value of *B* ensures this. The network **NET** is now said to satisfy the property of *BIO3G_Auth_Key* if all of its traces satisfy *BIO3G_Auth_Key*

$$\textbf{NET sat } \mathit{BIO3G\_Auth\_Key} \Leftrightarrow \forall\, tr \in traces(\textbf{NET}) \bullet \mathit{BIO3G\_Auth\_Key}(tr)$$

Definition 2. *BIO3G_User_Auth*(*tr*) =

$$tr' \,\hat{}\, \langle Commit.MO.USIM.B.K' \rangle \leqslant tr \Rightarrow \langle Running.UE.USIM.A \rangle \text{ in } tr'$$

$$\wedge\ \#(tr \upharpoonright Running.UE.USIM.A) \geqslant \#(tr \upharpoonright Commit.MO.USIM.B.K')$$

Observe that *MO* receives no direct data from the user *U* or *UE* – it only receives a single message from *USIM*. It does, however, receive the unique value of *B*, which can be used to confirm the validity of biometric sample *A* obtained from the user *U*. The derivation of a new key *K'*, such that $K'= f_3(K,B)$, and its subsequent use (note that we assume *K* is already shared between *MO* and *USIM*), allows *MO* to be assured of the validity of *U*'s biometric sample.

We use *Running.UE.USIM.A* to indicate *U*'s submission of a biometric sample *A* to *UE* in this run. We use *Commit.MO.USIM.B.K'* to indicate *MO*'s run with *USIM* (using *B* and *K'* both of which point to a valid biometric sample). The specification then requires every time a *Commit.MO.USIM.B.K'* event occurs it is preceded by a *Running.UE.USIM.A* event. As in *Definition 1*, this clause alone does not provide injective agreement and so we place a second clause, which does provide a one-to-one relationship between these two runs. In terms of user authentication, this means that every time the mobile operator validates a biometric sample for *U*, *U* should have taken part in that run of the protocol earlier. The network **NET** is said to satisfy the property of *BIO3G_User_Auth* if all of its traces satisfy *BIO3G_User_Auth*

$$\textbf{NET sat } \mathit{BIO3G\_User\_Auth} \Leftrightarrow \forall\, tr \in traces(\textbf{NET}) \bullet \mathit{BIO3G\_User\_Auth}(tr)$$

5.2 Proof Strategy

Observe that the definition of **NET** so far uses a *Medium* process to abstract away the entire communication medium between the user equipment containing the UMTS subscriber module, that is, *UE* and *USIM* and the mobile operator *MO*. We consider a specific run of the protocol and redefine **NET** such that

$$\textbf{NET} = ((UE(A)_{\{submit\}}\|_{\{accept\}} USIM(KSB,CK,IK,K)) \mid\mid\mid MO(KSB,CK,IK,K)) \parallel Intruder$$

where the *Medium* process is replaced with an *Intruder* process. This allows us to, firstly, represent an intruder with specific capabilities as discussed in Section 4.1 (we enhance these capabilities in Section 5.4 to model all the functionalities relevant to our analysis) and, secondly, give an intruder complete control of the communication medium. Verifying the network **NET** for correctness would then mean that the BIO3G protocol can withstand all the attacks that such an intruder may launch on this protocol.

Our proof strategy is then as follows. We intend to verify the trace specification in *Definition 2*, that is, every time *MO* authenticates a valid biometric sample *A* from *UE*, indicated by *Commit.MO.USIM.B.K'*, the biometric sample *A* has been provided to *UE*, indicated by *Running.UE.USIM.A*. To achieve this, we require the trace specification in *Definition 1* to hold. Now in order for us to check whether every *Commit.MO.USIM.B.K′* is preceded by a *Running.USIM.MO.B.K′* (see *Definition 1*), we restrict **NET** on the *Running.USIM.MO.B.K′* signal event and check whether the following signal event *Commit.MO.USIM.B.K′* is allowed to appear in the restricted **NET**. More formally,

$$\forall\, tr \in traces(\mathbf{NET} \underset{Running.USIM.MO.B.K'}{\parallel} Stop) \bullet$$

$$\mathbf{NET} \underset{Running.USIM.MO.B.K'}{\parallel} Stop \ \mathbf{sat}\ tr \upharpoonright Commit.MO.USIM.B.K' = \langle\rangle$$

Observe that *BIO3G_Auth_Key* is slightly stronger (see *Definition 1*) than what we check in the proof strategy above. This allows us to only verify non-injective agreement between protocol runs, that is, we do not check whether for every *Commit.MO.USIM.B.K′* there is a unique *Running.USIM.MO.B.K′* that preceding it. This is due to the rank function theorem that does not verify injective agreement.

5.3 Constructing a Rank Function for BIO3G

We now construct a rank function for BIO3G as per the proof strategy and evaluate the different conditions provided in the theorem to prove its correctness.

We identify the ranks on the message space for our **NET** and construct the rank function shown in Figure 4 below. We consider the identities of *UE*, *USIM* and *MO* to be possibly impersonated by the *Intruder* process (therefore known to the intruder). We exclude, however, the identity of *U* and assign it a non-positive rank. We denote the values of the four parameters *ksb*, *ck*, *ik* and *k* as *KSB*, *CK*, *IK* and *K* and, assume them to be perfectly shared between *USIM* and *MO*, for a particular run of the protocol and not available to an intruder – we assign all such values a non-positive rank. A derived key *K′*, established as a result of this particular protocol run, is also assigned a non-positive rank. We consider all other keys to be available to the intruder, along with other values of the parameters and therefore assigned a positive rank. We assign a non-positive rank to a biometric sample *A* since it is unique to *U*. Recall that the rank function theorem is defined in terms of general sets *R* and *T*. We assign *R* = {*Running.USIM.MO.B.K′*} and *T* = {*Commit.MO.USIM.B.K′*}. This corresponds to the proof strategy described in Section 5.2, where we need to check for the occurrence of *Commit.MO.USIM.B.K′* in **NET** restricted on *Running.USIM.MO.B.K′*. We assign *Commit.MO.USIM.B.K′* a non-positive rank and assign *Running.USIM.MO.B.K′* a positive rank. Moreover, due to the proof strategy, we assign the messages $KSB \oplus D$ and $|KSB \oplus D|_{IK}$ a non-positive rank as they are not supposed to appear in the restricted network $\mathbf{NET} \underset{Running.USIM.MO.B.K'}{\parallel} Stop$.

In the rank function shown above, we also consider ranks for some general cases:

- We assign a concatenation of two messages m_1 and m_2 a positive rank only if both m_1 and m_2 are of positive rank,
- we assign two messages exclusive-or'd with each other a positive rank if they both have the same rank, that is to say, they both have either positive or

non-positive rank. So, if they both have a positive rank it means they are both available to the intruder anyway or, if they both have a non-positive rank, then it means the exclusive-or'd message can be sent out on a public channel (without the intruder retrieving either of the messages),

- we consider the fuzzy extractor function f_{fe} and assign the output of a function $f_{fe}(m)$ a positive rank only if the input m is of positive rank. This means, if the intruder is in possession of only $f_{fe}(m)$ and not m, then it is impossible to retrieve m. This allows us to model f_{fe} as a one-way function as defined earlier in Section 3,
- we assign the MAC-I value $|m|_k$, for some m, a positive rank only if both m and the key k are of positive rank. So an intruder cannot generate any MAC-I value without possessing both the message and the key, and finally,
- we consider the function f_3 (part of the UMTS-AKA standard) and assign the output of a function $f_3(k,m)$ a positive rank only if both inputs k and m are available to the intruder.

$$\rho(A) = 0 \qquad \rho(KSB \oplus D) = 0$$

$$\rho(|KSB \oplus D|_{IK}) = 0 \qquad \rho(m_1.m_2) = min\{\rho(m_1).\rho(m_2)\}$$

$$\rho(\mathcal{U}) = \begin{cases} 0 \text{ if } u = U \\ 1 \text{ otherwise (including if } u = UE \vee u = USIM \vee u = MO) \end{cases}$$

$$\rho(ksb) = \begin{cases} 0 \text{ if } ksb = KSB \\ 1 \text{ otherwise} \end{cases} \qquad \rho(f_{fe}(m)) = \begin{cases} 1 \text{ if } \rho(m) = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\rho(|m|_k) = \begin{cases} 1 \text{ if } \rho(m) = 1 \wedge \rho(k) = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\rho(f_3(k,m)) = \begin{cases} 1 \text{ if } \rho(k) = 1 \wedge \rho(m) = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\rho(m_1 \oplus m_2) = \begin{cases} 1 \text{ if } \rho(m_1) = \rho(m_2) \\ 0 \text{ otherwise} \end{cases}$$

$$\rho(Running.USIM.MO.B.K') = 1 \qquad \rho(Commit.MO.USIM.B.K') = 0$$

Fig. 4. A rank function for the BIO3G protocol

5.4 Verifying BIO3G Using Rank Functions

Before we proceed to consider each of the conditions of the rank function theorem and check whether our rank function in Figure 4 satisfies them, we alter the '$\vdash$'

relation (see Section 4.1) to enhance the capabilities of the intruder. We add the following rules, where m, m_1 and m_2 are some messages and, k is some key

- $\{m_1,m_2\} \vdash m_1 \oplus m_2$
- $\{m_1 \oplus m_2, m_2\} \vdash m_1$
- $\{m_1 \oplus m_2, m_1\} \vdash m_2$
- $\{m\} \vdash f_{fe}(m)$
- $\{m,k\} \vdash |m|_k$
- $\{m,k\} \vdash f_3(k,m)$

Recall the rank function theorem from Section 4.2. We proceed to consider each condition **R1**-**R4** and check whether the rank function in Figure 4 satisfies them.

R1) $\forall\, m \in \boldsymbol{IK} \bullet \rho(m) > 0$

The set ***IK*** contains all the information that the intruder is aware of at the start of the protocol. We consider the intruder an insider, such that it is also a valid subscriber of *MO* and is in possession of a corresponding set of parameters that *U* is. We denote such parameters as *KSB'*, *CK'* and IK_I'. We also consider an initial key K_I that the intruder shares with *MO*. The set ***IK*** then includes all this information, $\boldsymbol{IK} = \{UE, USIM, MO, KSB', CK', IK_I', K_I\}$. Nothing in this set is of non-positive rank.

R2) $\forall\, \boldsymbol{S} \subseteq \mathcal{M}, m \in \mathcal{M} \bullet ((\forall m' \in \boldsymbol{S} \bullet \rho(m') > 0) \wedge \boldsymbol{S} \vdash m) \Rightarrow \rho(m) > 0$

This conditions checks whether a message of non-positive rank can be generated under the '⊢' relation from a set of messages of positive rank. None of the messages of positive rank (see Figure 4), let the *Intruder* generate any messages of non-positive rank. Both *U* and corresponding biometric sample *A* are considered to be out of reach of *Intruder*. The parameters *KSB*, *CK*, *IK*, *K*, *K'* are assumed to be perfectly shared between *USIM* and *MO* (cannot be retrieved by the intruder). None of this allows the intruder to generate $KSB \oplus D$ or $|KSB \oplus D|_{IK}$ or anything of non-positive rank.

R3) $\forall\, t \in \boldsymbol{T} \bullet \rho(t) \leqslant 0$

This condition requires none of the events in *T* to be of positive rank. The only event in set *T* is the event *Commit.MO.USIM.B.K'* of non-positive rank (see Figure 4).

R4) $\forall\, i \in \mathcal{U} \bullet User_i \underset{R}{\|} Stop$ **sat maintain** ρ

Recall **NET** from Section 5. We consider each of the processes in **NET**, restrict them on *Running.USIM.MO.B.K'* and check whether they **maintain** ρ. Since only *USIM(ksb,ck,ik,k)*) can perform *Running.A.B.NB*, the other two processes remain unaffected. The process *UE(A)* with a biometric sample *A* behaves as follows

$$\begin{aligned} UE(A) \;=\; & biosample?A \rightarrow \\ & Running.UE.USIM.A \rightarrow \\ & submit.UE.USIM!f_{fe}(A) \rightarrow Stop \end{aligned}$$

The process *UE(A)* is not restricted to pass on the value $f_{fe}(A)$ (that is of non-positive rank) onto the channel *submit*. But since the submit channel is a private channel between *UE* and *USIM*, the intruder cannot observe $f_{fe}(A)$. The process *UE(A)* succeeds to **maintain** ρ. *USIM(KSB,CK,IK,K)* is restricted on *Running.USIM.MO.B.K'*

$$USIM(KSB,CK,IK,K) \underset{Running.USIM.MO.B.K'}{\parallel} Stop = $$
$$\begin{aligned}&accept.USIM.UE?B \rightarrow\\ &\text{IF } b = B \wedge k = K' \text{ THEN } Stop\\ &\text{ELSE } Running.USIM.MO.b.k' \rightarrow\\ &send.USIM!MO!(ksb\oplus d, |ksb\oplus d|_{ik}) \rightarrow Stop\end{aligned}$$

Now if the restricted process $USIM(KSB,CK,IK,K) \underset{Running.USIM.MO.B.K'}{\parallel} Stop$ is passed on the value of B that leads onto the value of K', then it is instructed to stop. If, however, $b \neq B$, then the process continues to behave normally with appropriate values. Due to the restriction on the process, it does not transmit $KSB\oplus D$ or $|KSB\oplus D|_{IK}$ and therefore succeeds to **maintain** ρ. The process $MO(KSB,CK,IK,K)$ remains unaffected by the restriction.

$$\begin{aligned}MO(KSB,CK,IK,K) \;=\; &receive.MO?USIM?(KSB\oplus D, |KSB\oplus D|_{IK}) \rightarrow\\ &Commit.MO.USIM.B.K' \rightarrow Stop\end{aligned}$$

Upon observation, if $MO(KSB,CK,IK,K)$ does not receive the concatenated message $KSB\oplus D, |KSB\oplus D|_{IK}$ then it does not perform $Commit.MO.USIM.B.K'$. The only way it can perform $Commit.MO.USIM.B.K'$ is for it to receive $KSB\oplus D, |KSB\oplus D|_{IK}$. The process therefore succeeds to **maintain** ρ. All four conditions of them theorem **R1**-**R4** are satisfied by the rank function in Figure 4. This ensures that

$$\forall\, tr \in traces(\mathbf{NET} \underset{Running.USIM.MO.B.K'}{\parallel} Stop) \bullet$$
$$\mathbf{NET} \underset{Running.USIM.MO.B.K'}{\parallel} Stop \;\mathbf{sat}\; tr \upharpoonright Commit.MO.USIM.B.K' = \langle\rangle$$

We do, however, need to verify that **NET** also satisfies *Definition 2*. More formally,

$$\forall\, tr \in traces(\mathbf{NET} \underset{Running.UE.USIM.A}{\parallel} Stop) \bullet$$
$$\mathbf{NET} \underset{Running.UE.USIM.A}{\parallel} Stop \;\mathbf{sat}\; tr \upharpoonright Commit.MO.USIM.B.K' = \langle\rangle$$

Recall the definition of **NET**

$$\mathbf{NET} = ((UE(A)_{\{submit\}}\|_{\{accept\}}USIM(KSB,CK,IK,K)) \;|||\; MO(KSB,CK,IK,K)) \parallel Intruder$$

Let us focus on the component $((UE(A)_{\{submit\}}\|_{\{accept\}}USIM(KSB,CK,IK,K))$ that represents *UE* running alongside *USIM*. This component, due to its design, satisfies the condition that every time *USIM* receives the value B, *UE* has sent the value B (where $B = f_{fe}(A)$) to *USIM* prior to that. This represents the inner operations of the user equipment, where it is understood to function correctly and not leak any information to any other channel. For some trace tr of **NET**,

$$tr' \wedge \langle Running.USIM.MO.B.K'\rangle \leqslant tr \Rightarrow \langle Running.UE.USIM.A\rangle \text{ in } tr'$$

it satisfies

$$\forall\, tr \in traces((UE(A)_{\{submit\}}\|_{\{accept\}} USIM(KSB,CK,IK,K)) \bullet$$
$$tr' \frown \langle Running.USIM.MO.B.K' \rangle \leqslant tr \Rightarrow \langle Running.UE.USIM.A \rangle \text{ in } tr'$$

As the rest of **NET** cannot interfere with this property, the entire **NET** also satisfies

$$\forall\, tr \in traces(\mathbf{NET}) \bullet tr' \frown \langle Running.USIM.MO.B.K' \rangle \leqslant tr \Rightarrow$$
$$\langle Running.UE.USIM.A \rangle \text{ in } tr'$$

and since we have already proved

$$\forall\, tr \in traces(\mathbf{NET}) \bullet tr' \frown \langle Commit.MO.USIM.B.K' \rangle \leqslant tr \Rightarrow$$
$$\langle Running.USIM.MO.B.K' \rangle \text{ in } tr'$$

it follows that

$$\forall\, tr \in traces(\mathbf{NET}) \bullet tr' \frown \langle Commit.MO.USIM.B.K' \rangle \leqslant tr \Rightarrow$$
$$\langle Running.UE.USIM.A \rangle \text{ in } tr'$$

implying that every time *MO* authenticates *B*, the corresponding sample *A* was accepted by *UE* prior to that. □

A note on injective agreement. Observe that *Definition 2* is slightly stronger than what we have proved in Section 5.4. The second clause in *Definition 2* requires injective agreement between runs such that every time *Commit.MO.USIM.B.K′* appears, there is a unique *Running.UE.USIM.A* event preceding it. The rank function theorem does not verify injectivity so we cannot use rank functions to prove this. Note, however, that the design of the protocol ensures injective agreement in a number of ways:

- The integrity algorithm f_9 (see Section 3) uses *FRESH* and *COUNT-I* as parameters that prevent the MAC-I value of the message (and therefore the message) being replayed. The use of *FRESH* prevents any intentional replay on behalf of an intruder whereas *COUNT-I* prevents any confusion due to transmission errors,
- at the end of the protocol run, both *USIM* and *MO* derive a new key, which is derived using the existing key between the two parties and then used for subsequent communication. If for some reason *MO* is somehow led to accept a replayed message from a previous run, any subsequent communication would then not be possible as the newly derived key by *USIM* and *MO* would not match, and finally,
- recall the component $((UE(A)_{\{submit\}}\|_{\{accept\}} USIM(KSB,CK,IK,K))$ that represents *UE* running alongside *USIM*. We consider this component to work perfectly and specify its behaviour as

  $$tr' \frown \langle Running.USIM.MO.B.K' \rangle \leqslant tr \Rightarrow \langle Running.UE.USIM.A \rangle \text{ in } tr'$$

  for some trace *tr*. Due to the design of the component, every time *UE* sends the value of some *B* to *USIM*, it accepts it. Conversely, every time *USIM* accepts the value of some B, UE has actually passed B onto $USIM$. We can therefore refine the behaviour of this component further as, for some trace tr,

$$tr' \frown \langle Running.USIM.MO.B.K' \rangle \leqslant tr \Rightarrow \langle Running.UE.USIM.A \rangle \text{ in } tr'$$
$$\wedge \; \#(tr \upharpoonright Running.UE.USIM.A) \geqslant \#(tr \upharpoonright Running.USIM.MO.B.K')$$

that ensures injective interaction between UE and $USIM$ for this component.

The three mechanisms combine to ensure that BIO3G provides injective agreement between UE and MO. This is important not only for authentication purposes, but also for key establishment between the two parties. □

6 Discussion

The introduction of biometrics for security in 3G networks is a challenging process. BIO3G was created by following a design approach that identified the necessary requirements and defined the corresponding specifications, through the detailed study of biometric technologies within the framework of their incorporation in a 3G environment. The use of a single message makes the protocol less prone to cryptanalysis. The generation of non-invertible values for biometric samples is particularly important as it avoids actual biometric samples being sent over public channels. The use of the existing facilities of the *USIM*, for key generation and simple exclusive-or functions, makes BIO3G lightweight and compatible to the existing 3G infrastructures. These features, as well as the simplicity of the exchanged messages over the air interface, allow BIO3G to be ideal for mobile communications where computational resources are limited, the medium is noisy and ever increasingly hostile and, more importantly, actual biometric data is too risky to be stored on user equipment (whether stationary or mobile).

Schneider's rank functions approach is helpful in gaining an insight into the design of such protocols. The approach is also useful as it allows us to model algebraic properties of the message constructs, such as the use of exclusive-or to combine messages. Such a formal analysis raises confidence in the design of such protocols.

References

1. Neimi, V., Nyberg, K.: UMTS Security. John Wiley & Sons (2003)
2. Benoit, O., Dabbous, N., Gauteron, L., Girard, P., Handschuh, H., Naccache, D., Socile, S., Whelan, C.:Mobile Terminal Security. Cryptology ePrint Archive Report 2004/158
3. ISO/IEC JTC1, SC37/SG1: Biometric vocabulary corpus (2004)
4. Dimitriadis, C., Polemi, D.: Biometrics –Risks and Controls. Information Systems Control Journal (ISACA), vol.4 (2004) 41-43
5. Dimitriadis, C., Polemi, D.: A protocol for incorporating biometrics in 3G with respect to privacy. In Fernandez-Medina, E., Hernandez, J., Garcia, J. (eds.): Proceedings of the 3rd International Workshop on Security in Information Systems (WOSIS'05) 123-135
6. 3rd Generation Partnership Project: TS 33.102 - 3G Security; Security architecture (2004)
7. Hoare, C. A. R.: Communicating Sequential Processes. Prentice-Hall International (1985)

8. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy Extractors: How to generate strong keys from biometrics and other noisy data. Advances in Cryptology -- Eurocrypt 2004, Lecture Notes in Computer Science 3027, Springer-Verlag (2004) 523-540
9. 3rd Generation Partnership Project: TS 22.022 - Personalisation of Mobile Equipment (ME); Mobile functionality specification (2005)
10. Roscoe, A.W.: The Theory and Practice of Concurrency. Prentice-Hall (1997)
11. Schneider, S.: Concurrent and Real-time Systems: the CSP Approach. Addison-Wesley (1999)
12. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G. and Roscoe, B.: Modelling and Analysis of Security Protocols. Addison-Wesley (2001)
13. Schneider, S.: Verifying Authentication Protocols in CSP. IEEE Transactions on Software Engineering, 24(9), (1998) 741-758
14. Dolev, D and Yao, A.C: On the security of public key protocols. IEEE Trans. on Information Theory, 29(2), (1983) 198-208

Attribute-Based Authentication Model for Dynamic Mobile Environments[⋆]

Michael J. Covington, Manoj R. Sastry, and Deepak J. Manohar

Corporate Technology Group, Intel Corporation

Abstract. Rich, context-aware applications are emerging for mobile computing environments that provide new and innovative services to consumers. Security is critical for the successful realization of this vision. We propose a new method for authentication that utilizes contextual information to overcome the limitations inherent in traditional approaches. We have defined a model that uses contextual attributes to achieve an approach to authentication that is better suited for dynamic, mobile computing environments. We examine the use of trusted platforms to provide assurances for these contextual attributes. Our model promotes the adoption of many revolutionary mobile applications by providing a seamless and flexible user experience that can protect privacy and reduce administrative overhead.

1 Introduction and Motivation

Traditional approaches to authentication are not always well-suited for rich mobile applications because they fail to utilize contextual information present in the mobile environment. The traditional approach to authentication requires principals to present personal identity information to obtain access to services. We have proposed a new approach to authentication [1] that utilizes contextual attributes to achieve seamless authentication. In this paper, we extend our previous work by presenting an authentication model that utilizes trusted platform capabilities to achieve authentication using contextual attributes.

As users become more mobile and network connectivity becomes more ubiquitous, it is increasingly important that users and service providers not rely too heavily on the network infrastructure to provide certain security services. Network components that are owned by an independent–and potentially untrusted–party could easily lie to either the user or the service provider in an effort to compromise secrecy or privacy. Allowing the platform to report information pertaining to its own context places the client in control of its authentication data. Relying on the network opens up a new wave of potential attacks, such as a rogue access point that lies about the client's location in an effort to alter the results of the client's access request.

In attribute-based authentication, the contextual information is asserted; the authentication process must determine if that assertion is true. In other words,

[⋆] Copyright ©2006 Intel Corporation

attribute-based authentication requires the platform to provide assurances in the security of the contextual information that is being reported.

The remainder of this paper proceeds as follows: Section 2 introduces contextual attributes, discusses the limitations of traditional authentication, and investigates the role of trusted platforms for providing security assurances of contextual attributes. In Section 3 we present a detailed model for achieving attribute-based authentication. Section 4 explores a concrete usage scenario in which we utilize a trusted platform to achieve attribute-based authentication. We provide a discussion of the complications that are introduced by our model in Section 5. Finally, Section 6 looks at related work, and we conclude with Section 7.

2 Contextual Attributes

A significant body of work has emerged around the use of contextual information in computer systems. Context can be used to provide these systems with certain capabilities inherent to human perception and reasoning. As computers become more pervasive, interpreting context becomes an essential requirement for next-generation applications. Context describes a specific situation by capturing the setting in which an event occurs.

These observations have led us to consider the impact of context on security services. In particular, we are interested in how contextual attributes, the fundamental primitives that comprise context, can be used to support and enhance authentication in a dynamic, mobile environment. In this section, we explore the inherent limitations of traditional authentication techniques that do not utilize context. We then explore contextual attributes and examine how they can be used for security purposes. Through our research, we have identified some possible categories of contextual attributes and provide examples of each. In addition, we describe a usage scenario and explore the use of trusted platforms to collect security-relevant contextual attributes.

2.1 Limitations of Traditional Authentication

Authentication schemes typically fall into one of three categories for user verification: physiological, knowledge-based, and object-based. For a variety of reasons, these traditional methods of authentication may not be well suited or appropriate for a mobile computing environment. Primarily, these approaches are not seamless (some form of explicit user intervention is required to verify identity), they are not flexible (some administrative “setup” is required and users are often required to identify themselves completely, regardless of the sensitivity of the transaction), and they require that the user provide private user identity information.

In a dynamic mobile setting, contextual information is prevalent but currently unused. For instance, personal identity information is not always required for providing services to the user; instead, contextual information may alone be sufficient. With an abundance of information available to describe users and their operating environment, we propose an authentication method using contextual attributes.

2.2 Security-Relevant Contextual Attributes

Transactions in a mobile environment involve the user, the mobile platform, the specific resource or service being accessed, and the physical environment of both the user and platform. Contextual attributes can describe the user's mobile operating environment, without necessarily disclosing the user's personally identifiable information. In addition, contextual attributes allow policy administrators to specify flexible policies that can apply to a variety of access requests. For instance, instead of granting access to a set of user IDs, a policy administrator can grant access to any user matching a specific attribute (e.g., the user requesting access is in a certain location).

We observe some possible categories for security-relevant contextual attributes in a mobile environment. These include attributes associated with:

- the **physical environment**, such as time, temperature, etc.
- the **service** being accessed, such as "basic" service or "premium" service
- the **participants in a transaction**, such as a user's location
- the **platform**, such as a platform's trust state
- a specific **transaction**, such as an e-receipt, e-token, etc.

2.3 Role of Trusted Platforms

Contextual information is useful only when it is acquired and reported in a secure manner. Rich capabilities are emerging on the mobile platform that can provide contextual attributes. For example, cell phones now have location sensing capabilities. However, such information lacks security assurances. One limitation in achieving attribute-based authentication through existing mobile platforms is the lack of security assurances for acquiring and reporting contextual attributes. In the following paragraphs, we explore the role of trusted platforms for achieving seamless authentication.

The Trusted Computing Group (TCG) [2] is a standards organization that is actively working to specify the details for building a trusted platform. TCG has defined a platform independent security subsystem called the Trusted Platform Module (TPM). The TPM serves as the root of trust in a platform and provides a foundation for a challenger to make an assessment of the trustworthiness of a platform. In addition, the TPM also provides a mechanism called sealed storage that lets a user tie secrets to a particular platform and its state.

In our attribute-based authentication model, the client utilizes the properties of a trusted platform to gain access to services being offered by a service provider. Knowing that the client's request originated from a trusted platform provides security assurance to the service provider. Specifically, the service provider has assurance that the contextual attributes were securely collected and reported by the client platform. The service provider has assurance that the client platform has not been tampered and that the integrity of platform services responsible for collecting and reporting contextual attributes are intact. Overall, a trusted platform plays a significant role in attribute-based authentication by facilitating a client to convince the service provider that its request for service was legitimate.

3 Attribute-Based Authentication Model

In this section, we present a high-level usage scenario that motivates the need for contextual attributes in order to provide seamless authentication. We then propose a model to achieve attribute-based authentication. In Section 3.2, we identify the assumptions that were made in our model and then provide a detailed description of the model in Section 3.3.

3.1 Overview

Consider a scenario in which a coffee shop has partnered with a premium content service provider. The coffee shop has agreed to pay access charges to the service provider on the user's behalf for those who have made a purchase. Alice enters the coffee shop and makes a purchase. The traditional approach to authentication would require the coffee shop vendor to provide Alice a user name and a password in order to access the service. Alice would then have to enter this user name and password to gain access to a specific premium service.

By examining the same scenario, this time with attributes being used for authentication, we are able to see a number of benefits to using this approach. As depicted in Figure 1, Alice makes a purchase using e-cash and her mobile platform stores an electronic receipt confirming the purchase. The service provider requests Alice's platform for location information and amount of purchase recorded in the e-receipt. Alice's location and electronic receipt information are seamlessly

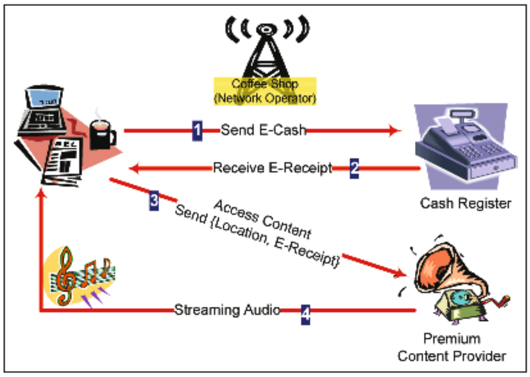


Fig. 1. Coffee Shop Scenario

transferred to the service provider as long as the policy on Alice's system allows this information to be disclosed. After verifying the location information and the amount of purchase, the service provider provides premium services to Alice. Based on this scenario, it can be seen that attributes seem to be a natural fit for rich mobile applications. Further, the use of attributes can allow a service provider to provide a rich variety of services tailored to the value of the attributes.

Authentication using attributes requires security assurance from the platform providing these attributes. Unlike the traditional authentication mechanism, the entity authenticated in this case is the platform.

3.2 Assumptions

We now describe the assumptions that were made as we developed our model for attribute-based authentication. These assumptions pertain to the authentication model, system requirements, and infrastructure components.

One of our key assumptions that drove the design of an attribute-based authentication model was that users would be highly mobile and that they may not have user accounts setup on all possible services that are available to them. Furthermore, because of a highly mobile user population, service administrators would find it hard to create user accounts for each and every user that can access the service. In our model the service administrator uses the entity's attributes to decide whether the entity can access the service or not.

In order to provide better integrity guarantees for reported attributes, we propose that all entities participating in our protocol run on trusted platforms. To support this, we also assume that the accompanying trusted platform infrastructure has been setup and is operational. Our model also relies on the existence of the infrastructure accompanying trusted platforms. We rely on the Privacy CA component of the trusted platform infrastructure to provide the attestation identity keys and some additional functionality which we describe in Section 3.3.

In the next section we describe the system model in detail. At a high level, we have entities that provide the attributes and entities that verify the attributes. Our protocol is not involved in establishing any form of trust relationships between the attribute-providing entities and the attribute-verifying entities. We assume that this happens via some out of band mechanism and for simplicity assume that once the trust relationship has been established they behave in a trustworthy manner.

When using temporal attributes we assume that the clocks of the participating entities are synchronized. All the entities that operate on a client's attributes are assumed to be trustworthy in the sense that they will not reveal the identity of the client. The client controls the attributes that are revealed to other entities, thus controlling how much of history is revealed.

3.3 System Model

Our model for attribute-based authentication is based on contextual information from the situation in which an access request is made. The model is comprised of the following logical components: a client, an attribute provider, the service provider, an attribute verifier, and a certificate authority. A high-level overview of our seamless authentication model is given in Figure 2. In this section, we detail the functionality of these components and their interactions.

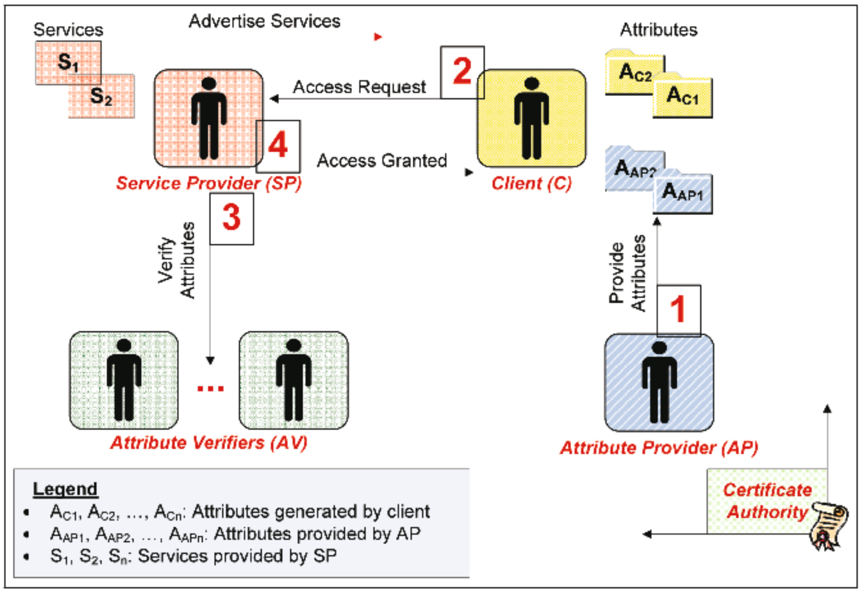


Fig. 2. Overview of Authentication Model

Client. In our model, the client (C) refers to an entity attempting to access a protected service or resource. This entity would be comprised of both the user and the user's platform. For the purposes of our model, the platform includes the hardware and software components that allow him to perform a specific function; these components serve an important role in the trust chain that exists between the user and service being accessed, thus we treat them as a collective unit.

The client platform is responsible for collecting relevant contextual attributes and reporting them to a service provider. Contextual attributes must be handled in a manner that provides a remote entity to have certain assurances regarding the data's integrity. Contextual attributes can be obtained by the client either directly from the environment or from an attribute provider.

Attribute Provider. The attribute provider (AP) is the entity that makes contextual information available to the client. A variety of attribute providers may provide similar contextual attributes, thus giving clients and service providers an option to choose their preferred provider.

As stated above, context describes a specific situation by capturing the setting or circumstances in which an event occurs. A contextual attribute represents a measurable contextual primitive; it is the full set of contextual attributes that comprise the context of a situation. Given this definition, a client may choose from a collection of APs to capture the situation in which he operates. For example, a variety of APs may provide current location coordinates to clients; it is up to the client to select a specific provider based on their needs. A GPS coordinate, for instance, is going to have a different meaning and perhaps quality than another type of location coordinate that may be available.

Service Provider. In our model, the authentication component resides with the Service Provider (SP). The service provider acts as an intermediary between the client and the protected services, and is responsible for verifying identity information that is passed to the authentication component. Services in our model (s_1, s_2, ..., s_n) are the protected resources that the client wants to access. In order to gain access to the service, the client must be authenticated by proving its identity to the service provider managing the requested services.

In traditional authentication, known identity information pertaining to the client is compared with information received from a source claiming to be that client. With contextual attributes, it is the responsibility of the authentication component to verify the source of those attributes. When provided by a client, the service provider must request that the client provide proof that its contextual attribute collection components were not compromised. Also, the service provider must determine if the attributes were obtained from an attribute provider. If a third-party was the source of the attributes, the SP must also determine if it trusts the AP.

In addition, the attributes being presented may not be known a priori. Instead, the type of information is known, but specific values are not. It is the responsibility of the authentication component to take the specific values provided and verify that they match the contextual attribute requested (e.g., location coordinates provided on behalf of a client must fall within the physical space that defines a coffee shop).

Attribute Verifier. The Attribute Verifier (AV) is responsible for verifying both the source of a contextual attribute and the attribute itself. In our model, the service provider passes contextual attributes that it cannot independently verify to an AV. The AV analyzes the attribute to confirm that it is from a valid source. The AV then repackages the attribute and sends it back to the Service Provider, with whom it has a pre-established trust relationship. The AV serves as a trusted intermediary that maintains relationships with both the SP and the AP. For example, an SP may utilize one AV to validate all location-related contextual attributes. This enables the SP to manage only one trust relationship for location attributes (with the AV); the AV is then responsible for staying up-to-date on all location technologies and attribute providers so it can serve the requests from the SP.

Certificate Authority. We have decided to use public key encryption via a Certificate Authority (CA) to aid with the distribution and verification of keys between components. Instead of generating any new CA infrastructure requirements, our model extends the existing functionality of the trusted computing infrastructure's Privacy CA. Typically, this CA is responsible for providing the client with an attestation certificate that is used to provide integrity information to an external challenger. We extend this functionality to include public identity certificates for the service provider. The service provider's public key is used by the client to encrypt the contextual attributes sent to the service provider. The Privacy CA must evaluate the client's platform and issue Attestation credentials that have to be trusted by the service provider in order for the service provider to verify attestation reported by the client.

3.4 Authentication Using Attributes

In this section, we describe the various interactions that take place between the entities described in our logical system model. We discuss three different authentication cases, starting with a simple interaction involving only the SP and C, and then expand this to include APs and AVs. We refer the reader to Figure 2 for a comprehensive overview of our authentication model; subsequent figures will provide details on the individual cases being presented.

We begin with service provider SP advertising available services to clients $C_1, C_2, \ldots, C_n$, as described in (a). For example, some Location-based Services (LBS) allow providers to advertise their services to any user within a certain proximity.

$$SP \xrightarrow{\text{Advertise Services}} C_1, C_2, \ldots, C_n \qquad \text{(a)}$$

The advertised service announcement could contain the following:

- **URL** - A Universal Resource Locator could be provided that would allow C to locate and access the advertised service
- **Authentication Requirements** - The SP may choose to advertise its authentication requirements in advance of the access request, thus allowing the C to package its access request with the necessary authentication credentials. For example, in the coffee shop scenario above, the broadcasted authentication requirements would request location and e-receipt information from the client.
- **Certificate** - SP may provide a signed certificate that allows the client to secure future communications with the service provider or any of its advertised services.

Case 1: Basic Authentication. Figure 3 details a simple protocol for achieving attribute-based authentication; this protocol exchange captures the primary steps involved as contextual attributes are presented on behalf of a client and authenticated by a service provider. Here, client C requests access to perform operation O on service s_1 from service provider SP as specified in (b).

If s_1 is a protected resource, C will be required to present the necessary identity credentials and supporting verification information in order to proceed. In an

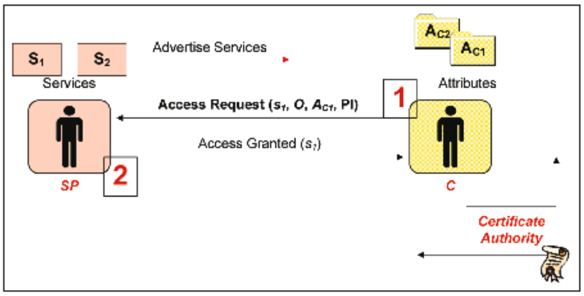


Fig. 3. Attribute-based Authentication (Case 1: Basic Authentication)

attribute-based authentication model, authentication requirements are dynamic and can vary dramatically from one access to the next. Given this, the client must collect the required contextual information and bundle it with the access request that is sent to SP. Additionally, C reports integrity information (using its trusted platform) and includes a nonce so that SP can verify trustworthiness of the client platform and ensure freshness of the request.

$$SP \xleftarrow{\text{Access Request (Service } s_1\text{, Op } O\text{, Attribute } A_{C1}\text{, Platform Integrity, nonce)}} C \qquad \text{(b)}$$

Contextual attributes must be trusted if they are to be used for authentication services as they can be easily spoofed or compromised. A trusted foundation allows SP to verify the integrity of the context-collecting platform. For instance, a client reporting the trusted location information can prove that the integrity of the platform that collected and reported the location attribute is intact.

The service provider now has a signed authentication package - containing the requested contextual attributes - that was provided on the client's behalf. The first step in attribute-based authentication for the service provider requires SP to determine the source of the attributes. This is achieved by verifying the cryptographic signature on the encrypted packet. If SP is able to determine that the attributes were provided by a trusted source, it will then determine if the reported attributes satisfy a certain criteria.

$$SP \xrightarrow{\text{Access Granted (Service } s_1\text{)}} C \qquad \text{(c)}$$

Once SP determines the authenticity of the attributes provided on behalf of C, it passes the authenticated credentials to the authorization engine that will complete the processing of the client's access request. Assuming that C is authorized to access the requested resource, SP responds with an "Access Granted" as indicated in (c).

Case 2: Attribute Providers. In a dynamic mobile environment, we do not expect that all attributes will be observed or generated solely by the platform. Instead, Attribute Providers will supply contextual information to those clients

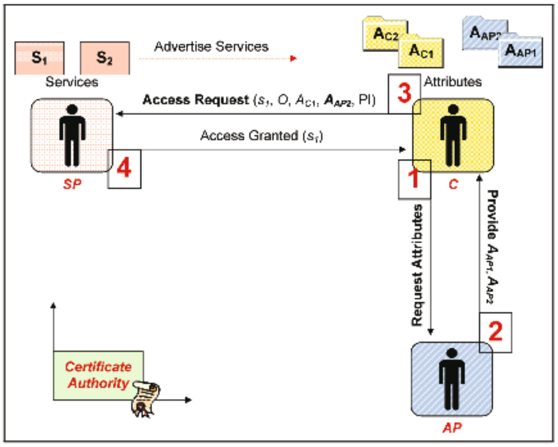


Fig. 4. Attribute-based Authentication (Case 2: Attribute Providers)

seeking to collect relevant information about the situation. For example, a variety of APs could supply location information to clients. A service provider analyzing such attributes would need to know the source of the contextual information in order to best determine its trustworthiness.

In Figure 4, we describe an expanded protocol to achieve attribute-based authentication using a third-party attribute provider. Similar to the protocol presented in Case 1, SP advertises services to C that, in turn, requests access to services maintained by SP. However, Case 2 provides the client with the option of collecting contextual attributes from a third-party AP. After requesting attributes from AP, as described in (d), the client obtains the relevant contextual attributes that can be used in its exchange with the service provider.

$$\begin{gathered} C \xrightarrow{\text{Request Attributes}} AP \\ C \xleftarrow{\text{Provide Attributes}(A_{AP2})} AP \end{gathered} \tag{d}$$

In this model, attributes can be acquired either directly through the client platform or from a third-party provider. In the previous case (Case 1), the client was unable to prove that the contextual information it observed from the physical environment was accurate or uncompromised.

$$SP \xleftarrow{\text{Access Request(Service } s_1\text{, Op } O\text{, Attribute } A_{C1}\text{, } A_{AP2}\text{, Platform Integrity)}} C \tag{e}$$

When an AP is introduced to the model, the access request itself must be altered to include information that was provided by the AP, as indicated in (e). SP must then determine the authenticity of the attributes provided by both C and AP before passing the authenticated credentials to the authorization engine.

Case 3: Attribute Verifiers. By adding APs to our model, we introduce a requirement on the service provider to establish trust relationships with all APs prior to any client interactions. In Figure 4, we overcome this limitation by introducing the Attribute Verifier (AV) which is responsible for managing trust relationships with various APs. For example, a service provider may employ a single AV to manage trust relationships with all location-based attribute providers.

$$SP \xrightarrow{\text{Verify } (A_{AP2})} AV$$
$$SP \xleftarrow{A_{AP2} \text{ Verified}} AV \tag{f}$$

The interaction (f) shows an SP requesting the AV to verify the validity of an attribute A_{AP2}. As providers are added and removed in the ecosystem, the AV can off-load the administrative overhead of managing these relationships from the service provider. As indicated in (f), the AV replies to the SP confirming the validity of passed attributes.

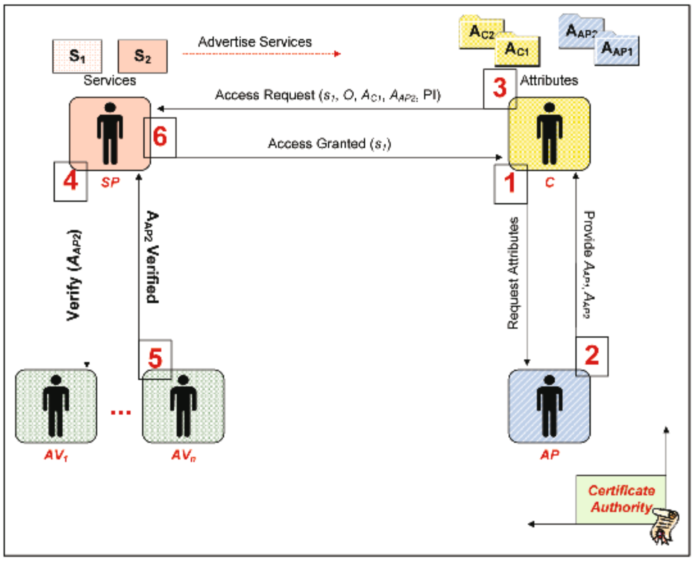


Fig. 5. Attribute-based Authentication (Case 3: Attribute Verifiers)

4 Usage Scenario

The traditional security model requires principals to present personal identity information to achieve authentication. However, there are situations in which the verification of specific user identity information is neither practical nor appropriate.

Consider the scenario previously discussed in Section 3.1. The coffee shop has agreed to pay access charges to the service provider on the user's behalf for those who have made a purchase and remain physically located in the coffee shop. In order to access these third-party services, Alice, a customer in the coffee shop, must provide proof that she is physically located in the coffee shop and that she has made a purchase.

In order to meet the purchase requirement, Alice enters the coffee shop and pays for her items using the electronic cash (e-cash) that is stored in a protected area on her mobile platform. She interacts with a digital cash register that accepts her e-cash and provides an electronic receipt (e-receipt) to indicate proof-of-purchase; the e-receipt is stored in a protected area on her mobile platform. Alice then requests access to the premium content service provider and verifies her "credentials" by providing location coordinates through her platform's location-reporting capabilities, along with a signed e-receipt. These contextual attributes are sufficient for the service provider to authenticate Alice's request as coming from a legitimate user of the content.

The service provider advertises premium services to the customers inside the coffee shop. Additionally, during service advertisement, the service provider lists the authentication requirements for each of its services. Alice requests access to a protected resource such as "Streaming Content" and, in addition, bundles this request with contextual attributes requested by the service provider during service advertisement. In this case, the contextual attributes include an e-receipt and current location coordinates. The service provider receives a signed package of attributes from Alice and proceeds to first verify if the source of these attributes is trusted and then checks to see if the reported attributes match the expected values. In this scenario, the service provider determines if the location information was supplied by a trusted platform or by an external source such as the service provider's access point physically located in the coffee shop. Upon successful authentication by the service provider, the request is passed on to the authorization engine to process the request. These contextual attributes are sufficient for the service provider to authenticate Alice's request as coming from a legitimate user of the content.

4.1 Benefits of Attribute-Based Authentication

We have identified several benefits associated with using attribute-based authentication. First, the authentication process is seamless. In the example above, contextual attributes can be provided on behalf of Alice by the mobile platform, thus alleviating her from the burden of providing verification information that can be presented by the platform.

Attribute-based authentication also protects private user identity information. This is achieved by using contextual attributes in place of private user identity information, when appropriate.

Attribute-based authentication is flexible in the sense that it allows users to control which attributes are revealed to a service provider. Since contextual attributes are collected and reported by the platform, the user remains in control of the attributes that are released to the various parties he interacts with.

Attribute-based authentication alleviates the burden of managing specific user identity information. For example, a premium content provider does not need to manage the user's personal identity information. In the coffee shop scenario, location and possession of an electronic receipt (or token) are the contextual attributes required for Alice's authentication to the premium content service provider; no other information (e.g., username/password) is required to authenticate Alice in this scenario. In this example of Seamless Authentication, Alice's attributes were sufficient for the service provider to grant access without requiring Alice's personal identifying information.

On the other hand, in cases where user identity information is absolutely required (e.g., during a sensitive transaction), contextual attributes can be used to provide stronger authentication by supplementing traditional techniques with an attribute-based approach. For example, knowing a specific user ID is helpful, but placing constraints on when that user can be authenticated using contextual attributes such as location and time may be helpful in some policies. We intend to explore this in future work.

5 Discussion

An attribute-based authentication model allows for users to be authenticated without any user account being setup a priori. In a highly dynamic computing environment, it is possible that the attributes used to authenticate a client to a service provider may change before the client completes accessing the protected resource. If the SP were to ignore the change in the client's attributes, it would be similar to existing systems where a user can continue his current session even if his account is deleted (after login) on the remote authentication server. If the SP wants to disallow access to the client immediately because the client's attributes have changed, we need a mechanism that monitors the contextual attributes. We don't discuss such a mechanism in this paper and defer it for future work.

Our attribute-based authentication model relies on the client to voluntarily pass some of its attributes to a SP. This potentially equates to the client revealing some of its personal history. In Section 3.2, we have identified a fundamental assumption behind our model: that the SP, AP and AV do not reveal the identity of the user. In an attribute-based authentication model, the user needs to be wary of rogue entities posing as SPs that simply collect a user's personal information.

Users should also be able to control the attributes that they are willing to share with different SPs. We believe a technique, similar to that described by the Platform for Privacy Preferences (P3P) Project [3], allowing users to write policies that govern how their attributes are used and with whom they are shared should be implemented. In a scenario similar to "Case 3" in Section 3.4, the SP would have to demonstrate that it meets a minimum set of requirements, as defined by the user, and that it also utilizes AVs that enforce a similar set of privacy policies.

Unlike existing authentication mechanisms, an attribute-based authentication model might have to verify more than just one attribute before it can grant access to a client. This provides an opportunity for a malicious client to launch a denial of service attack on an SP by sending a large number of forged attributes. To reduce the possibility of a denial of service attack in this manner, the attribute verification should not be computationally expensive. Our model allows for the SP to offload attribute verification to AVs.

6 Related Work

In this section, we briefly highlight existing research that has influenced our work with attribute-based authentication and trusted platforms. We discuss previous work with distributed authentication models, including parameterized authentication and constrained channels, as well as related work on context-aware authorization models and other notable research.

In our previous work [1], we have identified limitations inherent in traditional security methods when applied to a dynamic and mobile environment. We defined contextual attributes and investigated using these attributes to achieve seamless authentication. In addition, we listed the benefits of using contextual attributes to achieve authentication and proposed utilizing the properties of a trusted platform to securely handle contextual attributes.

Noble and Corner [4] describe "Transient Authentication" as a means of authenticating users with devices through a small, short-ranged wireless communications token that they wear. This research demonstrates the use of location (proximity) as an attribute in authentication. We have presented a more generic approach that allows any attribute to be used for authentication.

Covington et al. [5] have introduced a new model for user authentication called Parameterized Authentication that can be used to secure dynamic applications in pervasive computing environments. The major benefit provided by this authentication model is its ability to provide quantitative measure for an authentication parameter when faced with incomplete, inaccurate or unreliable information. At times, attributes other than a username and password combination may be more appropriate given the nature of the request.

Kindberg et al. [6] have introduced an approach to authentication that utilizes contextual information pertaining to the principal in place of his actual identity. Their approach, however, relies heavily on physical channel characteristics to determine context and lacks a generic method for context validation. Our approach using trusted platforms removes the dependency on network infrastructure and, instead, relies on the trusted platform to securely store and report security-relevant authentication attributes.

Creese et al. [7] present a general overview of security requirements for pervasive computing and discuss how traditional authentication does not fit these requirements. They thoroughly discuss authentication of entities using attributes, however, they do not present a model for authentication as we have done in this paper.

The notion of using attributes, such as location, with digital identities has been defined by the Liberty Alliance Project [8]. In our work, we have defined contextual attributes to capture a situation within which a user can be granted access to a restricted resource based *solely* on contextual attributes. These attributes include subject attributes (e.g., location), object attributes (e.g., "basic" content), transaction attributes (e.g., an e-receipt), and platform attributes (e.g., whether the system is trusted or not). These contextual attributes enable a service provider to offer a broad range of services based on the value of these dynamic attributes.

Giuri and Iglio [9] have proposed an enhancement to the Role-based Access Control (RBAC) model that provides special mechanisms for the definition of content-based policies. By extending the notion of permission, they have allowed for the specification of security policies in which the permissions to an object may depend on the content of the object itself. These role templates can be extended to specify constraints on subject roles, thus preventing users from achieving certain identities until specific environmental conditions are met.

Extending role templates to constrain subject roles can be achieved through Environment Roles, as defined by Covington et al. [10]. Similar to our definition of an attribute, environment roles were designed to capture the security-relevant context or state of the environment. Environment roles can be used to create role templates that specify constraints that must be bound to a subject role.

7 Conclusion

Rich, context-aware applications are emerging for mobile computing environments that provide new and innovative services. As consumers begin to rely on the mobile platform to perform sensitive transactions, it is increasingly important that the mobile platform provide higher security assurances.

The traditional approach to authentication is not always well-suited for a mobile environment. We have proposed a new approach for authentication that utilizes contextual information present in the environment, hence making it better suited for a dynamic mobile environment. In this paper, we describe an attribute-based authentication model that takes into account the contextual attributes pertaining to a specific situation in which an access request was initiated. We have identified assumptions, trust dependencies and described interactions between the various components of the authentication model. Further, we examined the role of trusted platform in providing security assurances for the contextual attributes used for achieving authentication. Our approach provides several benefits, including protected user privacy, seamless user authentication, and reduced administrative overhead for the service provider.

References

1. Sastry, M.R., Covington, M.J.: Attribute-based authentication using trusted platforms. In: Proceedings of the 8th International Symposium on Wireless Personal Multimedia Communications (WPMC '05), Aalborg, Denmark (2005) Special Session on Platform Security.
2. Trusted Computing Group: TCG specifications. TCG Website (2005) Available from: `http://www.trustedcomputinggroup.org/home/`.

3. World Wide Web Consortium (W3C): Platform for Privacy Preferences (P3P) Project (2005) Available from: `http://www.w3.org/P3P/`.
4. Corner, M.D., Noble, B.D.: Protecting applications with transient authentication. In: Proceedings of the 1st international conference on Mobile systems, applications and services. (2003) 57–70
5. Covington, M.J., Ahamad, M., Essa, I., Venkateswaran, H.: Parameterized authentication. In: Proceedings of the European Symposium On Research In Computer Security (ESORICS). (2004)
6. Kindberg, T., Zhang, K., Shankar, N.: Context authentication using constrained channels. In: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications. (2002)
7. S.J. Creese, M.H. Goldsmith, B.R., Zakiuddin, I.: Authentication in pervasive computing. 1st International Conference on Security in Pervasive Computing (SPC) (2003)
8. Liberty Alliance Project: Liberty alliance project specifications. Liberty Alliance Website (2006) Available from: `http://www.projectliberty.org/`.
9. Giuri, L., Iglio, P.: Role templates for content-based access control. In: Proceedings of the Second ACM Workshop on Role Based Access Control, Fairfax, Virginia, USA (1997) 153–159
10. Covington, M.J., Long, W., Srinivasan, S., Dey, A., Ahamad, M., Abowd, G.: Securing context-aware applications using environment roles. In: Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT), Chantilly, Virginia, USA (2001) 10–20

Author Index