Security in Pervasive Computing and Protocols for Protecting Privacy

Michael Østergaard Pedersen

Progress Report

Department of Computer Science
University of Aarhus
Denmark

December 2005

Supervisor: Ivan Damgaard

Contents

Chapter 1

Introduction

This report describes the work the author was part of during the first part of his Ph.D. studies, together with ongoing work and directions for the next two years.

The main research interests of the author is the development of secure protocols for protection of users privacy. This involves both coming up with new protocols and investigating how already published results can be used to solve real world problems. Another major research direction concerns security in pervasive computing, especially with regards to privacy.

The results obtained so far are described in this report. The first result is a protocol where a user can prove membership of a group without revealing his identity, unless the user cheats and gives his identifying information to someone else. We call this concept *unclonable group identification* [22].

On the pervasive computing side, the results include a report written for the former danish council of IT security [42] describing current and future issues regarding security in pervasive computing, where especially usability and privacy were found to be the most important problems to solve. Currently the author is part a group under the danish board of technology, which aims to describe the opportunities and threats of using RFID, and also to provide suggestions on how to solve some of the problems with RFID and privacy.

We have also published an article [7] at UbiComp 2003 about our work on solving the usability issue of logging into a system multiple times during the day at a hospital. Our solution is based on smart cards with cryptographic capabilities. A prototype of the system was implemented as part of a larger framework.

Furthermore the author is involved in designing the security architecture as well as looking at the software architecture side of the eu-DOMAIN platform [27] which is an ambient intelligence service platform for automatic and context sensitive offering and contracting of mobile web services across heterogeneous networks. The security architecture is currently being designed and implemented and regarding the software architecture we are in the early stages of writing an article about using domain models in ambient intelligence. We will not directly describe eu-DOMAIN any further in this

report. Primarily due to space constraints, but also since most of the work done by the author is not interesting from a research perspective. Still, work done on security in eu-DOMAIN has served as inspiration to future work and has highlighted issues regarding security in pervasive computing. Whenever relevant, these issues will be mentioned in the other sections.

Based on experiences within the area of pervasive computing, future work will focus on designing protocols to solve some security issues in that area.

The rest of the report is structured as follows. In chapter 2 we describe how a new efficient scheme for obtaining signatures on committed values can be used to provide various forms of privacy protecting protocols such as group signatures, anonymous credential systems and unclonable group identification. In chapter 3 we describe the security problems in pervasive computing as well as our results related to that area and finally, chapter 4 discusses future work.

Chapter 2

Protocols for Privacy Protection

In this chapter we describe how a new efficient scheme for obtaining signatures on committed values can be used to provide various forms of privacy protecting protocols such as group signatures, anonymous credential systems and unclonable group identification.

2.1 Introduction to Privacy

When trying to solve privacy in larger systems it is important to realise that not all privacy problems can be solved by cryptographic protocols alone, so before talking about protocols for protecting privacy, we should make clear what kind of privacy we want to protect.

Going back to basics we might ask ourselves: "What is privacy"? Probably the most well known treatment of privacy was done by Irwin Altman. He describes privacy as a *self-environment boundary regulation process based on dynamic, social, dialectic normalisation of desired privacy to attained levels* [2]. The definitions in [2] are very wide and many researchers have used them to examine more closely what the different dimensions of the privacy space are [10, 38, 41] in order to find out why we care about privacy, how privacy violations occur and how we can protect our privacy and the privacy of others.

In order to understand the privacy implications of systems, it is important to realise that there are different dimensions of the privacy space. When talking about privacy we are not talking about the whole privacy space, but rather some set of phenomena that can be classified into subspaces. Lederer and Dey [38] have identified the six dimensions seen in figure 2.1.

I will only briefly describe them here since my work is not about the definition of privacy. Simply put, privacy violations occur because someone (the observer) gains some information about another person (the subject) that the subject does not want

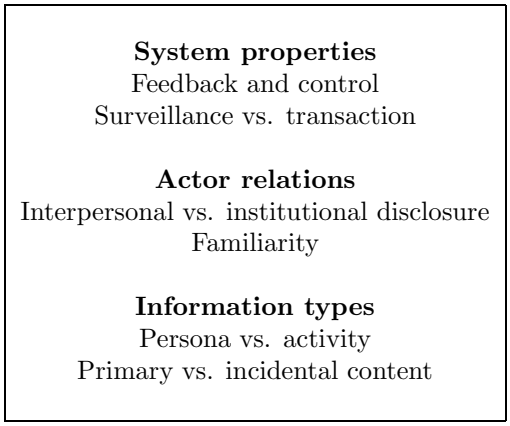

Figure 2.1: The six dimensions of the privacy space.

to observer to have.

One of the ways by which privacy phenomena differ is how much control the subject has over the disclosure of personal information and if the disclosure occurs by surveillance of the subject or transactions performed by the subject. This is the *system properties*. Another way to differentiate privacy phenomena is by the subjects relation to the observer. Is the observer a person or an institution and is the subject familiar with the observer. The third way in which privacy phenomena differs is by the type of information disclosed. For example there is a huge difference between disclosing information about the existence of a person or information about that person. Further, the disclosure of sensitive information can have different implications depending on whether that information was the primary or incidental content of the disclosure.

It is important to realize that since we are talking about information, privacy is strongly related to confidentiality. We might view privacy as confidentiality of personal information which gives us a link between traditional data security and privacy, and demonstrates that protocols and systems can indeed help to protect some dimensions of the privacy space.

2.2 Signature Schemes

Digital signature schemes play an important role in many areas. Not only for allowing users to sign documents electronically, but also for their use as building blocks to build more complex protocols. Digital signature schemes were invented by Diffie and Hellman [25], later formalised by Goldwasser, Micali and Rivest [30] and can be realised if and only if one-way functions exist [44]. These general constructions are not very practical, so over time, more efficient schemes have been designed. In the *random oracle* model (see figure 2.2) other, and more efficient, schemes can be proven secure, but since an ideal random function does not exist in the real world, these

schemes cannot be proven secure in the plain model.

Random oracle model	Assumes the existence of a random function mapping $\{0,1\}^* \rightarrow \{0,1\}^k$ for some fixed integer k, $k > 0$. In other words, when the random oracle is queried with an input value, it returns a uniformly random bit string of length k. If the query is repeated, the output string is the same.
Plain model	Assumes no random oracles or common parameters.

Figure 2.2: Definition of some cryptographic models.

Some schemes were proposed that were provably secure in the plain model, all based on the strong RSA assumption. In 2002 Camenisch and Lysyanskaya proposed a signature scheme based on the strong RSA assumption that could be used as a building block for other applications and in 2004 they proposed the first signature scheme that is provably secure in the plain model under the discrete-logarithm-based assumption [14]. This scheme can also be used as building blocks in many protocols, and since this is the scheme we have used, we will describe it in the following section.

In order for a signature scheme to be useful as building blocks for other protocols, it should:

- Provide protocols for obtaining a signature on a committed message
- Provide protocols for proving in zero-knowledge the possession of a signature on a committed message.

The following protocol has these properties as well as some additional ones, like unconditional hiding of the message to be signed even if the user obtains a signature on the same message, and proves knowledge of the same message, multiple times.

2.2.1 Signature scheme by Camenisch and Lysyanskaya

We start with some notation and preliminaries. We will use the notation from Camenisch and Stradler [15] for proofs of knowledge of discrete logarithms. For instance,

$$PK\{(i,j,k,l) : \chi = \alpha^i \beta^j \wedge \tilde{\chi} = \tilde{\alpha}^k \tilde{\beta}^l\}$$

denotes a zero-knowledge proof of knowledge of integers i, j, k, l such that $\chi = \alpha^i \beta^j$ and $\tilde{\chi} = \tilde{\alpha}^k \tilde{\beta}^l$ holds.

Assume that we have a *setup* algorithm that takes as input a security parameter 1^k and outputs two groups $G_p = \langle g \rangle$ and $\mathbf{G_p} = \langle \mathbf{g} \rangle$ of prime order $p = \Theta(2^k)$ that have a non-degenerate bilinear map $e : G_p \times G_p \rightarrow \mathbf{G_p}$. Bilinear means that for all $P, Q \in G_p$, for all $a, b \in Z, e(P^a, Q^b) = e(P, Q)^{ab}$ and non-degenerate means that there exist some $P, Q \in G_p$ such that $e(P, Q) \neq 1$, where 1 is the identity in G_p.

This can be used to generate a signature scheme in the following way:

Key generation Let $(p, G_p, \mathbf{G_p}, g, \mathbf{g}, e)$ be the output of the setup algorithm. Choose $x \in Z_p$ and $y \in Z_p$ and set $X = g^x$ and $Y = g^y$. Then the secret key is $s_k = (x, y)$ and the public key is $p_k = (p, G_p, \mathbf{G_p}, g, \mathbf{g}, e, X, Y)$.

Signature On input message m, secret key s_k and public key p_k choose a random $a \in G_p$ and output the signature $\sigma = (a, a^y, a^{x+mxy})$.

Verification On input message m, public key p_k and purported signature $\sigma = (a, b, c)$ check that the following equations hold

$$e(a, Y) = e(g, b) \quad \textit{and} \quad e(X, a) \cdot e(X, b)^m = e(g, c)$$

Proof that this is actually a signature scheme can be found in [14]. For this to be useful as building blocks we need a obtain a signature on the message m without revealing m. Note that if we give g^m as input to the signer instead of m the algorithm still works if we choose $\sigma = (g^r, a^y, a^x M^{rxy})$, because $a^x M^{rxy} = a^{x+mxy}$. Of course the user needs to prove knowledge of m in order for the signature scheme to remain secure.

Second, we need a way to prove possession of a signature in zero-knowledge. First we blind the signature σ by choosing random $r, r' \in Z_p$ and setting $\tilde{\sigma} = (a^{r'}, b^{r'}, c^{r'r}) = (\tilde{a}, \tilde{b}, \tilde{c}^r) = (\tilde{a}, \tilde{b}, \hat{c})$ and then send this blinded signature to the verifier. Both parties now compute $v_x = e(X, \tilde{a})$, $v_{xy} = e(X, \tilde{b})$ and $v_s = e(g, \hat{c})$ and then carry out the following proof:

$$PK\{(\mu, \rho) : v_s^{\rho} = v_x v_{xy}{}^{\mu}\} \tag{2.1}$$

The verifier accepts if this proof is correct and it holds that $e(\tilde{a}, Y) = e(g, \tilde{b})$. One should note that this scheme (called scheme A in [14]) does not unconditionally hide the message m because the value g^m will be the same if the user tries to obtain a signature on the same message. This is solved by signing an unconditionally hiding commitment to the message instead. Some modifications is made to the scheme in order to allow this. This modifies scheme is called scheme B. Scheme C is a generalisation of scheme B that allows signing blocks of messages $(m^{(0)}, m^{(1)}, ..., m^{(t)})$. Below is a description of scheme C.

Key generation Let $(p, G_p, \mathbf{G_p}, g, \mathbf{g}, e)$ be the output of the setup algorithm. Choose $x \in Z_p$, $y \in Z_p$ and for $1 \leq i \leq t, z_i \in Z_p$. Set $X = g^x$, $Y = g^y$ and for $1 \leq i \leq t, Z_i = g^{z_i}$. The secret key is $s_k = (x, y, z_1, ..., z_t)$ and the public key is $p_k = (p, G_p, \mathbf{G_p}, g, \mathbf{g}, e, X, Y, \{Z_i\})$.

Signature On input message $(m^{(0)}, m^{(1)}, ..., m^{(t)})$, secret key s_k and public key p_k choose a random $a \in G_p$ and set $A_i = a^{z_i}$ for $1 \leq i \leq t$, $b = a^y$, $B_i = (A_i)^y$ for $1 \leq i \leq t$, $c = a^{x+xym^{(0)}} \prod_{i=1}^{t} A_i^{xym^{(i)}}$. Output the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$.

Verification On input message $(m^{(0)}, m^{(1)}, ..., m^{(t)})$, public key p_k and purported signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ check the following:

1. $e(a, Z_i) = e(g, A_i)$

2. $e(a, Y) = e(b, g)$ and $e(A_i, Y) = e(g, B_i)$
3. $e(X, a) \cdot e(X, b)^{m^{(0)}} \prod_{i=1}^{t} e(X, B_i)^{m^{(i)}} = e(g, c)$

Exactly as in scheme A, the signer doesn't need to know the message. The only place where the message is present is in the computation of c in the signature, but if the user instead of $m^{(0)}, m^{(1)}, ..., m^{(t)}$ sends a commitment on the form $M = g^{m^{(0)}} \prod_{i=1}^{t} Z_i^{m^{(i)}}$ then we can still compute $c = a^x M^{axy} = a^{x+xym^{(0)}} \prod_{i=1}^{t} A_i^{xym^{(i)}}$.

If $t = 0$ this scheme is equivalent to scheme A and with $t = 1$ it is equivalent to scheme B. Like for scheme A, there exist a protocol for proving possession of a signature in zero-knowledge. First we blind the signature σ by choosing random $r, r' \in Z_p$ and setting $\tilde{\sigma} = (a^{r'}, \{A_i^r\}, b^{r'}, \{B_i^r\}, c^{r'r}) = (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \tilde{c}^r) = (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c})$ as in scheme A and send it to the verifier, but this time in addition to computing v_x, v_{xy} and v_s, both parties also compute $V_{(xy,i)} = e(X, \tilde{B}_i)$ and then carry out the following proof protocol:

$$PK\{(\mu^{(0)}, ..., \mu^{(t)}, \rho) : v_s^\rho = v_x (v_{xy})^{\mu^{(0)}} \prod_{i=1}^{t} \left(V_{(xy,i)}\right)^{\mu^{(i)}}\} \tag{2.2}$$

The verifier accepts if this proof holds and $e(\tilde{a}, Z_i) = e(g, \tilde{A}_i)$, $e(\tilde{a}, Y) = e(g, b)$ and $e(\tilde{A}_i, Y) = e(g, \tilde{B}_i)$. The latter ensures that the A_i's, B_i's and c were formed correctly.

2.3 Anonymous Credential Systems

A credential system is a system where a user can get access to some resource by presenting a credential showing that he is authorised to do so. In the real world, credentials are often drivers license or passports, but in the digital world it will often take form of a digital signature. An anonymous credential system consists of users and organisations, where organisations only know the users by pseudonyms. Different pseudonyms of the same user cannot be linked, yet a user can prove possession of a credential issued to a pseudonym to another organisation where the user is known by a different pseudonym, without revealing any more than the fact that the user has a credential. Possession of a credential can be demonstrated an arbitrary number of times and these demonstrations cannot be linked to each other. It should be practically impossible for users to forge credentials even if they cooperate. Also even if organisations team up, they should not be able to find out anything about a user, in particular different pseudonyms belonging to the same user cannot be linked.

Anonymous credential systems, also called pseudonym systems, were first introduced by Chaum [17] and in order to realise them it is sufficient to have a commitment scheme, a signature scheme and protocols for [39]

1. Proving equality of two committed values
2. Obtaining a signature on a value without revealing this value to the signer
3. Proving knowledge of a signature on a committed value

(2) and (3) follow from the signature schemes above, whereas (1) can be realised if we use the Pedersen commitment scheme [43].

2.3.1 A Simple Anonymous Credential System

In this section we will show how scheme B can be used to construct a simple anonymous credential system. Such a system will have two operations: *Grant* which grants a credential to a user and *Verify* which allows an organisation to verify the credential. This simple system will not have the notion of pseudonyms and will also only allow an organisation to issue one kind of credential. Furthermore this credential will not be able to contain attributes.

The general idea is the following: The user U chooses a secret key K which will be his identity, and a credential issued by an organisation O is a signature on this secret key.

First run the *setup* algorithm for the signature scheme to obtain the groups, the public and the secret key.

Grant U chooses a secret $K \in Z_p$ and $r \in Z_p$ at random and sends $M = g^K Z_1^r$ to O. O signs (K, r) and returns the signature $\sigma = (a, A_1, b, B_1, c)$ to U.

$Verify$ U and O carries out the proof protocol 2.2 and furthermore O checks that $e(\tilde{a}, Z_1) = e(g, \tilde{A}_1)$, $e(\tilde{a}, Y) = e(g, b)$ and $e(\tilde{A}_1, Y) = e(g, \tilde{B}_1)$.

Security of this system follows from security of the signature scheme as well as the unconditional hiding property of the Pedersen commitment scheme. The reason that this signature scheme is particulary suited for digital credentials is that it is easy to compute a different signature on the same message.

2.3.2 More Advanced Features

Besides handling pseudonyms as described above there are other interesting properties that would be nice to have in a credential system. One of them is for a credential to store attributes. For example a user could have a credential stating year of birth, current address, social security number, etc and then being able to prove that his age is greater than 18 without revealing any other information. Using scheme C the organisation can encode these attributes in the exponents and proof of knowledge of a signature can be extended to prove properties of these exponents, for example:

$$PK\{(\mu, \rho) : v_s^\rho = v_x {v_{xy}}^\mu, \mu \geq 18\} \tag{2.3}$$

Proving this is not trivial, but can be done by making a commitment c_μ to μ using an integer commitment scheme [23], proving that the value committed to in c_μ is equivalent to the value contained in 2.3 and finally use c_μ to show that the value committed to is greater than 18 [9].

A complete credential system with all these and more properties was developed by Camenisch and Lysyanskaya [12].

2.4 Group Signatures

To illustrate the use of this signature scheme as a building block, We will describe how it can be used to build group signature schemes. A group signature scheme is a scheme that allows members of a group to sign messages anonymously on behalf of the group, but when a dispute arises a designated revocation manager can revoke the anonymity of a group member.

A group signature scheme can be constructed from signature scheme A, an encryption scheme that is secure against adaptively chosen ciphertext attacks and a protocol for proving that a committed value is contained in a ciphertext. Such a scheme consists of the following five procedures:

1. A key generation procedure that produces a public key for the group as well as some secret keys for the group- and revocation manager
2. A join protocol that allows users to join the group
3. A sign protocol that allows members of the group to sign messages
4. A verification protocol that can verify that a message was indeed signed by a member of the group
5. A protocol the revocation manager can use to reveal the identity of a signer

We will describe how these procedures can be implemented. This idea is taken from [14].

Key generation The public key of the group manager is the public key of signature scheme A, that is $pk_{GM} = (p, G_p, \mathbf{G_p}, g, \mathbf{g}, e, X, Y)$ and his secret key is $sk_{GM} = (x, y)$. The public key of the revocation manager is the public key of the Cramer-Shoup encryption scheme [21] in the group $\mathbf{G_p}$, that is $pk_R = (\mathbf{h}, \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3})$ where $\mathbf{h}$ is chosen at random in $\mathbf{G_p}$, $\mathbf{y_1} = \mathbf{g}^{x_1}\mathbf{h}^{x_2}$, $\mathbf{y_2} = \mathbf{g}^{x_3}\mathbf{h}^{x_4}$ and $\mathbf{y_3} = \mathbf{g}^{x_5}$, where the secret key of the revocation manager is $sk_R = x_1, ..., x_5 \in Z_p$.

Join The user chooses a secret key $k \in Z_p$, sets $M = g^k$, sends M to the group manager and proves knowledge of k. The group manager replies with a scheme A signature (a, b, c) on k and stores $\mathbf{P} = e(M, g)$ along with the identity of the user.

Sign To sign a message m the user computes $\mathbf{P} = \mathbf{g}^k = e(M, g)$ and a blinded version of the signature $(\tilde{a}, \tilde{b}, \hat{c})$ as in the previous sections. Next the user encrypts $\mathbf{P}$ under pk_R which means choosing $u \in Z_p$ at random and setting $\mathbf{c_1} = \mathbf{g}^u, \mathbf{c_2} = \mathbf{h}^u, \mathbf{c_3} = \mathbf{y_1}^u\mathbf{P}$ and $\mathbf{c_4} = \mathbf{y_2}^u\mathbf{y_3}^{uH(\mathbf{c_1}||\mathbf{c_2}||\mathbf{c_3})}$. The group signature consists of $((\tilde{a}, \tilde{b}, \hat{c}), (\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4}), \Sigma)$ where Σ is the following proof-signature made using for example the Fiat-Shamir heuristic [28]:

$$\Sigma = SPK\{(\mu, \rho, \nu) : v_s^\rho = v_x v_{xy}{}^\mu, \mathbf{c_1} = \mathbf{g}^\nu, \mathbf{c_2} = \mathbf{h}^\nu, \mathbf{c_3} = \mathbf{y_1}{}^\nu\mathbf{g}^\mu, \mathbf{c_4} = \left(\mathbf{y_2}\mathbf{y_3}{}^{uH(\mathbf{c_1}||\mathbf{c_2}||\mathbf{c_3})}\right)^\nu\}(m)$$

Verify Validate Σ and check that $e(\tilde{a}, Y) = e(g, \tilde{b})$.

Open The revocation manager decrypts $(\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4})$ and obtains $\mathbf{P}$ which identifies the group member.

This scheme is secure in the random oracle model assuming the DDH assumption holds in $\mathbf{G}$ and it's easy to see why: Breaking the anonymity of a user requires breaking the encryption, since $(\tilde{a}, \tilde{b}, \hat{c})$ are just random values and Σ is derived from an honest verifier zero-knowledge proof. If it is possible to produce a signature that cannot be opened as belonging to a specific user, then the adversary can use rewinding to extract a forged signature, thereby breaking the signature scheme.

2.5 Unclonable Group Identification

We have used the signature scheme from [14] to construct a protocol to provide what we have called unclonable group identification. The following sections will contain an introduction to this concept, the motivation for doing this and a description of the protocol.

2.5.1 Introduction and Motivation

The problem of verifying that a given user is a member of a certain group, while ensuring that the user's personal identity is not revealed has been the focus of much research. Particular instances of this include group signatures [4, 14, 37] and anonymous credential systems [12] as we have already seen some examples of. However, in some applications a dishonest user has an interest in giving away some information that allows another user to identify as him as a member of the group - such as passwords or secret keys. The security problems implied by such a scenario have not been given much attention so far in the literature. It has been suggested to discourage this identity sharing by forcing users to either give away all their information or nothing, but here we are interested in cases where a dishonest user does in fact want to give everything away.

As a motivating example, consider the issue of software protection: It is well known that one of the strongest motivating factors in getting people to register as software users is if this enables some functionality that cannot be accessed without registration (and payment). This works particularly well, if the functionality requires access to the vendor's website, since then unauthorized access to the functionality cannot be achieved only by reverse engineering the software. In the case of games, for instance, the opportunity to play against others may be available to only registered users, and only through the vendor's website.

Verifying that a user is registered may be done in many different ways. We want a solution that would protect the users privacy, but at the same time doing as much as possible to protect against attacks where a user "clones" himself by handing a copy of his personal data (passwords, secret key(s), etc.) to another person in order to get the benefits of two registrations while only paying for one. Furthermore we want a solution where a user registers once and can then connect anonymously an unlimited number of times with the same key material.

A simple observation is of course that we can only detect a cloning attack if the user and his clone actually connects to the vendors website. Another simple observation is that we cannot detect a cloning attack where the user connects, leaves and then the clone connects, since the server can't distinguish between the user and his clone. What we *can* hope to detect, however, is if both the user and the clone connect so that they are on the site simultaneously, since this is exactly what cannot occur if the user has been honest. Not only do we wish to detect this attack, but we also wish to learn the identity of the user who cloned himself so we can either punish him or ban him from the system. This scenario is also of practical relevance. For instance, the case of a user who buys one copy of a game and distributes it to all his friends so they can play against each other online, is exactly a case where a number of clones would want to be connected simultaneously.

An *unclonable identification scheme* is an identification scheme where honest users can identify themselves anonymously as members of a group, but where clones of users can be detected and have their identities revealed if they identify themselves simultaneously. We have constructed a protocol that solves this problem.

First we looked at existing primitives to see if they could be used to solve the problem. One idea is to use group signatures [4, 14, 37] where the users identify themselves by signing a message chosen by the group manager, for example his current system time. This gives the users anonymity, but does not protect against cloning. If we want to protect against cloning, we would need a group signature scheme where it was possible to find out if two different signatures on the same message were from the same user. This does not follow from the standard definition of group signatures, and is actually false for known schemes, since these are probabilistic and produce randomly varying signatures even if the message is fixed.

Another possibility is to use E-cash schemes [18, 11] where each user receives some electronic coins and uses them to "pay" for access to the site. Cloning would be equivalent to double spending which can be detected even if the users are not logged on at the same time, but there are some issues with this. First of all, users would have to come back at regular intervals for more coins which would reveal information about how often they connect. Second, the cloning attack can not be detected if the user is careful not to spend the same coins which his clone is going to spend.

We have designed a protocol that solves this problem based on the signature scheme by Camenisch and Lysyanskaya [14].

2.5.2 Protocol Description

In this section I will give an overview of the protocol and the ideas behind it. For a full description with all the details, see our paper [22].

Preliminaries

There are two main ingredients in our protocol. The first ingredient is the signature scheme by Camenisch and Lysyanskaya and the second ingredient is a new technique for proving that an element in a group is of form g^{ψ} where ψ is a pseudorandom value computed from a committed key.

The protocol runs in "phases" where each phase is uniquely identified by a number α. An honest user must prove his membership of the group at most once in each phase. If he does it more than once in the same phase, he reveals his identity to the group manager. If he fails to prove his membership when required, he will be disconnected. However, if the user clones himself, then in order for both the user and his clone to stay connected at the same time, they must eventually both prove membership of the group during the same phase, which is equivalent to the same user trying to prove his membership of the group twice during the same phase.

First we introduce some general tools. Consider a group G_p of prime order p where $p = 2q+1$ and q is a prime. Let G_q denote the unique subgroup of Z_p^* of order q. Consider the case where where a prover knows exponents $x_1, ..., x_t \in Z_p$ such that $\beta = \alpha_1^{x_1} \cdots \alpha_t^{x_t}$ for publically known $\beta, \alpha_1, ..., \alpha_t \in G_p$. We are interested in allowing a prover P to prove knowledge of the x_i's to a verifier V. That is, we want:

$$PK\{(x_1, ..., x_t) : \beta = \alpha_1^{x_1} \cdots \alpha_t^{x_t}\} \tag{2.4}$$

The following Σ-protocol can be used to prove this:

1. P chooses $r_1, ..., r_t \in Z_p$ uniformly at random and sends to V $\tau = \prod_{i=1}^{t} \alpha_i^{r_i}$.
2. V chooses a random challenge $\epsilon \in Z_p$.
3. P responds with $z_i = r_i + \epsilon x_i \bmod p$ for $i = 1..t$. V checks that $\prod_{i=1}^{t} \alpha_i^{z_i} = \tau\beta^{\epsilon}$.

Recall the special soundness property of Σ-protocols: If the protocol is executed twice with the same first message τ and different challenges ϵ, ϵ', the verifier can extract the prover's secret x_i's. Imagine that the x_i's contain the prover's identity and that the first message τ will always be the same in the same phase α of the protocol. This will allow the verifier to learn the identity of a dishonest user.

Now consider the following change to the protocol. Instead of choosing the r_i's at random in the first message, let P choose them according to a pseudorandom function $\Psi_K(i, \alpha, b)$, where $K \in Z_q$ is a secret key known only to P, α is a public input, i is a number and b is a bit. For implementing this pseudorandom function we choose $\Psi_K(i, \alpha, b) = H(i, \alpha, b)^K$ where H is a hash function that outputs elements in G_q. This is a strong pseudorandom function in the random oracle model, assuming that the DDH assumption holds in G_q. The only problem is that this function outputs values in G_q where the r_i's were chosen in Z_p. This can be resolved by letting the exponents be chosen as the difference between two pseudorandom values which allows us to hit all of Z_p. The modified Σ-protocol looks like this:

1. P sets $r_i = \Psi_K(i, \alpha, 0)$ and $s_i = \Psi_K(i, \alpha, 1)$ and sends to V $\tau = \prod_{i=1}^{t} \alpha_i^{r_i - s_i}$.
2. V chooses a random challenge $\epsilon \in Z_p$.
3. P responds with $z_i = r_i - s_i + \epsilon x_i \bmod p$ for $i = 1..l$. V checks that $\prod_{i=1}^{t} \alpha_i^{z_i} = \tau\beta^{\epsilon}$.

We must allow P to show that he has followed the specific algorithm for constructing the r_i's and s_i's. The first step is for P to commit to these values. Let η, λ be random elements of G_p fixed in advance. P will make commitments $com_i = \eta^{r_i}\lambda^{\omega_i}$ and $com'_i = \eta^{s_i}\lambda^{\omega'_i}$, for $i = 1..t$ and random ω_i, ω'_i. We can now ask P to prove that he committed to the correct values, that is, execute

$$\begin{aligned} PK\{(r_i, s_i, \omega_i, \omega'_i, i = 1..t) : \tau &= \prod_{i=1}^{t} \alpha^{r_i}(\alpha^{-1})^{s_i}, \\ com_i &= \eta^{r_i}\lambda^{\omega_i}, com'_i = \eta^{s_i}\lambda^{\omega'_i}, i = 1..t\} \end{aligned} \tag{2.5}$$

The Σ-protocol for this is a standard variant of the one presented above. Assume that P has committed to his secret key K in some non-standard way, namely as $d = g^{\gamma^K \delta^r} h^u$ for publically known $g, h \in G_p$ and $\gamma, \delta \in G_q$. P now executes the following proof

$$\begin{aligned} PK\{(K, r, u, \omega_i, \omega'_i, i = 1..t) : d &= g^{\gamma^K \delta^r} h^u, \\ com_i &= \eta^{\Psi_K(i,\alpha,0)}\lambda^{\omega_i}, \\ com'_i &= \eta^{\Psi_K(i,\alpha,1)}\lambda^{\omega'_i}, i = 1..t\} \end{aligned} \tag{2.6}$$

which proves that he has used the committed key to generate the r_i's and s_i's. A Σ-protocol for this can be found in our paper.

A Protocol for Unclonable Group Identification

We will now show how a protocol for unclonable group identification can be realised.

First we run the setup algorithm from section 2.2.1 with a small modification, namely that we choose $p = 2q + 1$ where q is also a prime and then we let G_q denote the unique subgroup of Z_p^* or order q. The public key is, as the name implies, public and the group manager keeps the secret key. The group manager also chooses $\eta, \lambda \in G_p$ and $\gamma, \delta \in G_q$ to be used for the commitment scheme.

When a user U registers with the group manager, he makes a commitment $c_U = \gamma^K \delta^{r_U} \bmod p$ to a random secret key $K \in Z_q$, sends that commitment to the group manager and proves knowledge of K using the standard protocol for proving knowledge of discrete logarithms:

$$PK\{(K, r_U) : c_U = \gamma^K \delta^{r_U}\}$$

The group manager signs c_U using scheme A and returns the signature $\sigma = (a, b, c)$ on c_U to U. The value c_U is considered the user's identity to the group manager whereas (K, r_U, a, b, c) serves as the membership certificate.

For proving membership of the group, U must prove knowledge of a signature generated by the group manager on a value known to him, here c_U. This is the same as doing the proof of knowledge of a signature. Using the notation from section 2.2.1 this means proving:

$$PK\{(c_U, \rho) : {v_s}^{\rho} = v_x v_{xy}^{c_U}\}$$

We can use the modified Σ-protocol for proof 2.4 to prove this. In this proof we have, after adapting the notation, the following first message:

$$\tau = v_{xy}^{r_1 - s_1} v_s^{r_2 - s_2}$$

where the r_i's and s_i's are chosen as:

$$r_1 = \Psi_K(1, \alpha, 0), r_2 = \Psi_K(2, \alpha, 0), s_1 = \Psi_K(1, \alpha, 1), s_2 = \Psi_K(2, \alpha, 1)$$

U must now commit to these values, like this:

$$com_1 = \eta^{r_1}\lambda^{\omega_1}, com_2 = \eta^{r_2}\lambda^{\omega_2}, com_1' = \eta^{s_1}\lambda^{\omega_1'}, com_2' = \eta^{s_2}\lambda^{\omega_2'}$$

and prove that they are correct with respect to τ using the Σ-protocol for proof 2.5. This means executing the following proof:

$$\begin{aligned} PK\{(r_1, s_1, r_2, s_2, \omega_1, \omega_1', \omega_2, \omega_2') : \tau &= v_{xy}^{r_1 - s_1} v_s^{r_2 - s_2}, \\ com_1 &= \eta^{r_1}\lambda^{\omega_1}, com_2 = \eta^{r_2}\lambda^{\omega_2}, \\ com_1' &= \eta^{s_1}\lambda^{\omega_1'}, com_2' = \eta^{s_2}\lambda^{\omega_2'} \end{aligned}$$

Finally U must prove that each commitment contains a pseudorandom value of the correct form. Since $v_x^{-1} = v_{xy}^{c_U} v_s^{-\rho} = v_{xy}^{\gamma^K \delta^{r_U}} v_s^{-\rho}$ is a commitment to K this means proving:

$$\begin{aligned} PK\{(K, r_U, -\rho, \omega_1, \omega_1', \omega_2, \omega_2') : v_x^{-1} &= v_{xy}^{\gamma^K \delta^{r_U}} v_s^{-\rho}, \\ com_1 &= \eta^{r_1}\lambda^{\omega_1}, com_2 = \eta^{r_2}\lambda^{\omega_2}, \\ com_1' &= \eta^{s_1}\lambda^{\omega_1'}, com_2' = \eta^{s_2}\lambda^{\omega_2'}\} \end{aligned}$$

This can be done using the Σ-protocol for proof 2.6.

Once during each phase, the group manager will look for runs of the protocol where the first messages τ are identical. When this occurs he can use the special soundness property to extract c_U which will be the identity of a cheating user.

One problem is that since the user is completely anonymous, even though we catch him cheating in one phase, we have no way of identifying that user in a later phase. This can be solved using dynamic accumulators [13]. Simply put a dynamic accumulator is a hash function where you can add and remove values from the hash. By looking at the hash you learn nothing about which values are stored in the accumulator, but there is a zero-knowledge proof to prove that a given value is in the accumulator, without revealing that value. All it requires is that we have a commitment to the value we want to prove is in the accumulator.

We could extend our protocol in the following way: When a user registers and sends c_U as his identity, the group manager adds it to the accumulator. In each phase

we would require a user to prove that c_U is in the accumulator. This is fairly easy since we already have a commitment to c_U in the protocol, namely v_x^{-1} which is all the group manager needs to check this. If the user's identity is not in the accumulator he is denied access. A cheating user will reveal c_U and then the group manager can remove c_U from the accumulator.

Chapter 3

Security and Pervasive Computing

3.1 Introduction to Pervasive Computing

Pervasive computing deals with a near future where computers are everywhere, which is made possible by the technological development whereby chips are becoming smaller and cheaper all the time. Applications of pervasive computing could be things such as swivel chairs in offices that remember height settings, intelligent fridges that know whether or not the milk is too old, under-floor heating that is automatically turned on when the forecast is for cold weather, jackets with in-built mp3 players, sports clothing with in-built heart rate monitors, glasses with built in display devices and intelligent work clothes.

Today there is much focus on a few areas where we will see pervasive computing first. In the healthcare sector we might find things like bandages that can report how the injury is doing, video conferences with doctors, so patients can be treated at home and doors that open automatically, so the rescue services can enter the home of a patient with insulin shock. In the retail sector there are a large number of examples such as checkouts without cashiers, where the shopping trolley is pushed through a scanner that registers the products, smart product labels that can be used for providing a warning to people with allergies or for sorting waste and much more. For defence applications we will see things like unmanned planes, tanks and operating rooms, weapons that can only be fired by the rightful owner, etc. All these applications are based on many new units with built in computers and communication capabilities, of which we have probably only seen the tip of the iceberg.

According to Norman Cohen pervasive computing has a number of distinguishing characteristics [33]:

- Computing is spread throughout the environment and yet gracefully integrated with it.
- Users, devices, and services are often mobile.

- Information appliances are becoming increasingly available.
- Communication is made easy between individuals, between individuals and things, and between things.

The most prominent representative today is the RFID tag, which is a small passive devices with an antenna, that will transmit a unique number when it is in the proximity of a reader, but the list of pervasive computing technologies that is in the public domain is already long, and includes such things as mobile phones, PDAs and much more.

3.2 Security Problems in Pervasive Computing

In this section we look at some of the security problems we have identified in pervasive computing as well as some solutions people have been working on.

3.2.1 Background

In 2004 we were asked by the former danish council of IT security to do a report about security in pervasive computing. This report, finished later in 2004 [42], is a non-technical approach focusing on identifying problems instead of providing solutions in forms of specific recommendations regarding technologies to use.

The analysis was done based on scenarios describing a possible use of computers in the near future. Some of them were taken from the eu-DOMAIN project [27] whereas others were taken from an EU think-tank that tried to imaging how computers would affect our daily lives ten years from now. Based on these we classified the scenarios into three groups called ID-in-everything, services-in-everything and agents-in-everything. In short, ID-in-everything considers the use of devices that can only provide an identifier, such as RFID tags. Services-in-everything considers devices that in some way assist the user in his daily actions. Today PDAs, mobile phones, digital cameras, etc. are examples that would fit into this category. Agents-in-everything are the more futuristic scenarios where software agents loaded with our personal preferences are helping us in our daily lives, making decisions on our behalf, ordering food, negotiating contracts, etc.

A security analysis aims to identify threats and rate them based on how serious the threat is. This is done according to how serious the consequences of the realisation of the given threat is, and the likelihood of the threat happening. Such an analysis depends heavily upon the technology used and empirical studies, but since our work is based on future technologies and imagined use, such input is unavailable to us. The risk assessment was therefore done based on the current use of different technologies as well as a qualified guess as to how serious a given threat might be.

A threat is always made by some *actor* against an *asset*. As described in OCTAVE [1] a threat can be characterised by:

- The value of the asset
- Access to the asset (physical or logical)

- The actor threatening the asset
- The motive of the actor
- The outcome of the threat if it is realised by the actor

It would be tedious to characterise every single threat according to this, but nevertheless these items were still used as the background for our analysis.

When it comes to technology that will be everywhere, one type of actor deserves special attention and that is the regular user. Threats by the regular user are not described in the scenarios, but often the regular user is the source of security failures because they do not adhere to the security policy, for example by writing down his password on a piece of paper and leaving that piece of paper on his desk. Protecting against this problems requires education of the users, but more importantly, systems that are easy to use. If the user was able to log on to the computer simply by sitting down in front of it, he probably wouldn't have written down his password. In the near future we will no longer be able to choose not to use computers, which makes usability an even more important issue.

3.2.2 Identified Problems

We will briefly discuss the security problems of these three categories as one. Primarily due to space constraints, but also because they are in fact very similar. The more futuristic scenarios have some additional security problems, but the most important and relevant ones are already present in the two first groups of scenarios. We will look at which threats there are to the three security categories *confidentiality*, *integrity* and *availability*. For a full discussion of the security issues, see the report [42].

Confidentiality There are basically two types of confidential information in the scenarios: (1) The identity of some device and (2) the data present on, or generated by, a device. Generally all threats to confidentiality resulting from (1) are privacy violations which by many has been described as the greatest problem of pervasive computing. To protect against privacy violations occurring from disclosure of identity, there are the following three options:

- Anonymity (or confidentiality of identity)
- Protection against location tracking (or confidentiality of location)
- Protection against data aggregation (or combination of otherwise harmless information)

In many situations (e.g. with RFID tags) physical objects are assigned an identity, usually represented as a number. This identity number is then used to map a physical object to a database with more information about that object. This gives us three threats against privacy: Reading identity numbers from objects, abusing existing databases containing mappings between numbers and objects and a combination of the above. Since these identity numbers are often transmitted over wireless channels without any security, they are easy to obtain.

A RFID tags will reveal its identity number to any reader in the proximity of the tag, a GSM phone transmits its IMEI number when roaming between base stations, a laptop with a wireless network card transmits a unique identifier every time it sends out a packet on the network, etc. This problem is enhanced by the fact that the range from which this identity number can be read is far greater than the normal communication distance of the device.

An item with a RFID tag bought in a store can be used to track the customer around in the store as well as around the city. In most cases tracking an unknown RFID tag will only provide information about the location or activity, but by data aggregation it is possible the persona behind the activity can be disclosed as well. This kind of threat usually comes from cooperations that use this to track the behaviour of customers, their shopping preferences, etc. This kind of privacy violations normally cause users to distrust systems or organisations, and in the long term can neither be in the interest of the organisation or the users.

With access to the database mapping identity codes to information about the object they are attached to, privacy violations can be realised by people without the ability to read identity numbers from devices on the street, as long as they have access to the database. Probably the best example is the supermarket loyalty programs, where the supermarket keeps the entire shopping record of its customers in a database, along with timestamps of when they did their shopping. They don't need to be present in the supermarket to abuse that information (even though they most likely are in this scenario).

Combining the two scenarios above gives an even greater risk of privacy violations. Just reading some identity number once, will allow you to find more information by looking it up in the database. Taking the previous example, just reading some random identity number from a shopping bag on the street gives you information about who is carrying the bag, what else he is carrying, where he lives, where he did his shopping, how often he does it, how much money he spends in that specific supermarket, etc.

Many people respond to this threat by refusing to use new technology, but the technology has good uses as well, so simply not using it is not in our best interest as a society. The important thing is that the user should be in control of his own personal information and that he can rely on technology to protect his information, and not a privacy statement of a corporation.

Most of the current talk about revealing identity numbers of devices are related to RFID, but there are many other technologies with similar problems, and here designing a good and privacy protecting protocol on top of them might not be enough. Lower layers in the protocol stack might leak information such as the MAC address of network card, IP address, Bluetooth ID, location of the user based on GSM triangulation and signal strength, etc.

If we look at devices with more capabilities than just supplying an identity number there are additional threats. First we have the risk of disclosing information through transactions performed by the device. Since personal data are

not only stored in a database, but on the device itself, theft of the device might also disclose personal information. In case of medical data there might even be laws stating how these data must be protected.

Integrity When RFID tags are used to identify objects it becomes a problem that they are easy to spoof or replace with other tags, for example if the RFID tag is used as a price tag. If they are used to identify people, abuse of them can easily lead to identity theft.

With devices acting as sensors we must trust that the data we receive actually comes from the right sensor, so in some cases we need strong authentication from a device to a reader, but also from a user towards a device since unauthorised access to a device could violate the integrity of that device or data on it. We might also need a reader to authenticate towards a device so the device knows who it is sending information to.

An interesting problem regarding authentication is that we will sometimes have the scenario where a user authenticates towards a device and the device must somehow authenticate as the user towards another device. If the device that the user authenticates to directly is compromised, how much trust can the second device have in the authentication? Even more interesting: Can we somehow make the authentication works even if the device is compromised and what are the requirements for this?

Availability Some RFID protocols are highly vulnerable to a denial of service attack by a tag that tries to simulate all possible tags at the same time [35]. If RFID tags are used as anti-theft devices they can be placed in special designed packages that prevent a reader from communicating with the tag. This has in fact a dual purpose, as it is also a low-tech way to prevent the problems related to confidentiality of identity numbers regarding RFID tags.

For more capable devices running out of battery poses a significant problem. With hundreds of small devices at home it is infeasible (and expensive) to monitor their battery level and replace batteries as necessary. Most small devices are made to preserve battery power by going into sleep mode when not in use, but in some cases it is possible to deprive these devices of sleep by sending data to them, effectively reducing their battery life from a couple of years to a few days or even hours. Losing password or access tokens to a system is also a denial-of-service risk, which is a real issue when more and more non-technical users start using this technology. In case of a central device controlling other devices, malfunction of that device will have far reaching consequences for all attached devices, and measures must be taken to make it easy to replace such a device if it should fail.

What is hidden from the discussion above, is the role that the regular user plays, but it is mentioned between the lines. Systems for only disclosing identity to selected parties, authenticating based on a password, etc. must all be well controlled by the user, otherwise the technical solutions have no merit. With the number of small devices we expect people to buy and use in the future, security features must be easy

to use, cause no inconvenience for the user and must be enabled by default. To see what happens if this is not the case, check the current status on WiFi security. Also, these devices are small computers and will inherit all the security problems personal computers have today such as malicious code, viruses, worms, etc. if we are not careful.

The key to user acceptance of pervasive computing is to solve the privacy issues surrounding it, and where privacy can be achieved by technical means, it must be designed so even non-technical users get the benefits of privacy. In the literature it is generally assumed that there is always a trade-off between usability and security. While this is certainly true in many systems, we do not believe that is has to be true a priori. Privacy is not the only security issue for pervasive computing, but it is the most important one for the regular users of these systems and devices.

A related topic which has not been mentioned so far, but is relevant especially for the agents-in-everything scenarios, is the issue of trust. If we are expected to let software agents with access to personal data interact with foreign agents we have never interacted with before, there must be some way of deciding what level of interaction we will allow. This is not an issue we have looked at, but it has been studied in other projects such as SECURE [45].

3.2.3 Previous Work

Most of the security issues in pervasive computing are not different from the well-known problems of today. Authentication, protection of confidential data and ensuring the integrity of transmitted data are all known problems, but these problems might not be the same in all pervasive computing scenarios. Take authentication as an example. There is a difference between a scenario where one organisation certifies devices and only devices from that organisation are allowed to communicate with each other, and a scenario where devices must be able to communicate between different administrative and organisational entities that shares no information.

Another issue is the devices that must be supported. Many of them will be small devices without the computational resources to perform some of the heavy calculations needed for e.g. public key cryptography, and even if it is possible to equip these devices with such capabilities, many will refrain from doing so due to cost issues. Also they will have limited, if any, input devices, making seemingly trivial tasks such as typing in a password impossible.

There are no rule on how pervasive computing should be implemented, although most literature that deals with security in pervasive computing are centred around the scenario of small devices forming ad-hoc networks and communicating peer-to-peer. This was, however, not the case in the projects we have worked on. In the EPCiR project [31] a home is equipped with a central component called a gateway which takes care of all communication between devices and the outside world. In eu-DOMAIN [27] we also have gateways, but here gateways are only managed from a central place where all gateways are connected. Others have discussed the notion of a PAN (Personal Area Network) where a central component on a person (such as a PDA) handles communication with nearby devices and provides connectivity to the rest of the world. All of these architectures have their security problems and

somewhat different requirements to the solutions.

The rest of this section is about some problems with security in pervasive computing that various people have been working on.

If we want to do secure communication we need some cryptographic key material, so of course some research has been done on how to distribute key material. Anderson et. al. [3] suggest looking at a different threat model for wireless sensor networks. Normally, in security analysis, it is assumed that an attacker can eavesdrop on all communication going on, but actually this attacker rarely exist in the real world where many devices spread over a relatively small area (such as a house) need to agree on key material. They propose a scheme where a device starts by broadcasting a symmetric key with low power in the clear. If it doesn't receive a reply from another device, it increases the power and tries again. This is called whisper-mode and is supposed to limit the range from where the key can be eavesdropped. When all devices have agreed on a key with one of their neighbours, they start mixing their keys. Assume we have three nodes A, B, and C and three keys K_{AB}, K_{AC} and K_{BC} which are used to protect communication between respectively AB, AC and BC. If A wants a stronger key between himself and B he asks B and C to negotiate a new key based on the previous key between A and B in such a way that if K_{AB} was secure before, then the new key K'_{AB} is also secure, but if K_{AB} was compromised, then K'_{AB} is still secure if K_{AC} and K_{BC} are secure. If this is done at regular intervals by all devices, an adversary has to be able to monitor all traffic to keep track of the keys. If he misses just one key-exchange message, then after a short while he will no longer have access to any keys in the network.

Another possibility is key pre-distribution, which simply means that the keys must be put on the devices before they are deployed and can for example be used in combination with the previous method.

Another idea is to use channels that are authentic and hard to eavesdrop [6]. This could for example be by requiring physical contact between devices to do the key-exchange.

Stajano [46] proposed a method where the device gets a key from the first device it receives data from when it is turned on. This method is called the resurrecting duckling principle since it mimics the fact that a duckling will accept the first living being it sees as it's mother. Vaudenay describes a protocol [48] where a short and short-lived authenticated string can be used to generate a larger secure key that remains secure even if the short key is guessed.

Despite these results, there is still some way to solving all practical security problems regarding key exchange in pervasive computing. Relying on limited transmission range is not that useful. It has been shown with both WiFi and Bluetooth that with the right equipment it is possible to eavesdrop on the wireless communication from far greater distances than what the normal operational range is. WiFi has an operational range of around 100 meters, yet has been eavesdropped from a distance of 201 km [24]. Bluetooth has a range of around 10 meters, but has been eavesdropped from over 1000 meters away [19]. Also the solutions by Anderson et. al. are not suited for all types of scenarios such as when you are in a hostile environment and need a

secure communication channel between a few devices. Since symmetric keys cannot be used to authenticate individual devices this type of key exchange also has other limits. It is more suited to sensor networks than to scenarios involving authentication of individual devices. The resurrecting duckling principle also has some problems. It is easy for an adversary to setup a device with a stronger signal so the new device will recognize it as its mother duck. How many have tried using the neighbours WiFi access point by mistake? Still, many of the ideas from the resurrecting duckling principle are indeed useful. Key pre-distribution works, but how to do it in a user-friendly manner is another issue. Also if many devices need to communicate in an ad-hoc network using symmetric keys, they must each store a key for each device in the network which can be a problem for some devices. Finally authentic and secure channels are hard to come by, and using physical contact might not always be a feasible solution. A good solution will probably combine one or more of these approaches with good usability.

Since the most talked about issue regarding pervasive computing today is the issue of RFID tags, it should come as no surprise that some research has also been done in that area, most of it on authentication of RFID tags and readers. Obviously the best way to avoid leaking the identity of a RFID tag to every reader it passes, is to authenticate the reader first, but since RFID tags are very limited in computational resources this is no trivial task. An interesting observation is that RFID tags share the same lack of computing power with another computationally weak device, namely human beings. Therefore it has been suggested to use human authentication protocols for RFID tags [36]. Such a protocol called HB+ was proven secure, but under a threat model that was too restricted, and was later broken [29]. In another article [26] an authentication protocol is proposed based only on XOR and hash functions. Not only is this protocol useful for authentication, but can also be used to delegate ownership (and thereby the ability to read information from) the RFID tag to other parties. For example a store can transfer ownership of a RFID tag to the customer when he buys the item, instead of killing the tag and preventing the customer from taking advantage of the benefits of having RFID enabled goods at home. Unfortunately key management and usability can become a serious issue here since it is based on symmetric keys. Using pseudonyms as a model for privacy control with RFID tags is discussed in [34] and is based on anonymous credential systems, but has none of the features present in these systems. Another protocol for delegating control over a RFID tag that also uses pseudonyms to provide anonymity is proposed in [40]. The idea of allowing the user to control his own privacy is further studied in [5] which gives a practical model for user control of privacy based on rules and regulations, the type of data being protected and the users personal privacy preference. The last one is actually worth keeping in mind since different users might desire different levels of privacy.

3.3 Context-Aware User Authentication

In this section we present a brief overview of our paper titled *Context-Aware User Authentication - Supporting Proximity-Based Login in Pervasive Computing"* [7]. For a full description of our solution, including the authentication protocol and security

analysis, we refer to the paper.

3.3.1 Background

In our paper titled *Context-Aware User Authentication - Supporting Proximity-Based Login in Pervasive Computing"* we explore computer security in pervasive computing with focus on user authentication. We implemented a solution based on smartcards for user authentication and a context-awareness system for verifying the users' location. We also implemented a fall-back strategy in case the context-awareness system was down.

As mentioned before, there is often an inherent tradeoff between usability and security. User authentication mechanisms tend to be either secure, but less usable, or very usable, but less secure. It was our aim to try and combine the two standpoints and suggest a *context-aware user authentication mechanism* that is very usable as well as sufficiently secure for use in settings, where security matters, like a hospital environment.

User authentication can be based on three things. Something the user *has* (e.g. a smartcard), something the user *knows* (e.g. a password), or something the user *is* (i.e. a physiological trait) [47]. Traditionally if you combine some of these things, you get a more secure authentication mechanism than by just using one of them. Our design of a user authentication mechanism is based on supplementing well-known user authentication mechanisms with knowledge about the location of the user.

In a study of the use of Electronic Patient Records (EPR) at a large metropolitan hospital, we observed a number of usability problems associated with user authentication [8]. The EPR was accessed through PCs distributed within the hospital, and it had a traditional login system with usernames and passwords. Thus, whenever a clinician should access patient information he had to log in and out on different PCs, so it was not uncommon for a nurse to log in 30 times a day. Because this was a highly cumbersome thing to do in a hectic environment, workarounds were established. For example, users would avoid logging out, enabling them to return to the PC without logging in later; passwords were shared among users and made very easy to remember ('1234' was the most used password at the hospital); and users would often hand over user sessions to one another, without proper logout and login. Hence what might seem like a sufficiently secure system, was turned into a highly insecure system, because of obvious usability problems.

Even though this EPR system can in no way can be termed as pervasive technology, our study of its use has highlighted how essential user authentication is to the design of pervasive computing support for medical work in hospitals.

The Centre for Pervasive Healthcare [16] is involved in actively designing and developing pervasive computing technologies for use in hospitals. A central component in this effort is a basic runtime infrastructure, which supports Activity-Based Computing (ABC) [20]. The basic idea of activity-based computing is to represent a user's (work) activity as a heterogeneous collection of computational services, and make such activities available on various stationary and mobile computing equipment in a hospital. Clinicians can initiate a set of activities, and access these on various devices in the hospital. For example, a nurse can use the computer in the medicine

room to get some medicine, and later when giving this medicine to the patient she can restore the patient and medicine data on the display in the hospital bed. Central to this is clearly that users need to be authenticated on every device they want to use, and easy login is hence a core challenge in the concept of activity-based computing. If login was a problem with standard PC's that would be used 30 times a day, it is sure to be an issue here as well.

3.3.2 Requirements for a Pervasive Computing User Authentication Mechanism

Based on existing research within pervasive computing, our studies of medical work, and our experimental design effort with end-users, we can list the following requirements for a user authentication mechanism in a pervasive computing environment.

- *Proximity-based* – Work at a hospital is characterised by busy people who are constantly moving around, and are engaged in numerous activities in parallel. Easy and fast login was thus deemed a fundamental prerequisite for the success of a distributed, pervasive computing infrastructure, embedded in walls, floors, tables, beds, etc. The usability goal in our workshops reached a point where the user should do nothing to log in – he should simply just use the computer, and the computer would know who he was.

- *Secure* – Clinical computer systems store and handle sensitive, personal health data for many patients. It is therefore of utmost importance that these systems are protected from unauthorized access. Hence, pervasive computer systems in a healthcare environment require secure user authentication.

- *Active gesture* – We experimented with a login mechanism that would automatically transfer a user's on-going session to a display near him – much like the 'Follow-me' application using the Bat system [32]. This, however, turned out to be a less useful design. The problem was that often a clinician would enter a room, where numerous computing and display devices would be available. For example in a radiology conference room, there would be several wall-based displays, a wide range of desktop computers, and an interactive table where images can be displayed and manipulated. It was unclear from monitoring the location of the user, which of such displays he would like to use, or whether he wanted to use a computer at all. Therefore the authentication mechanism must be based on an active gesture near the display or devices that the user wants to use.

- *Support for logout* – During our experiments we discovered that the process of logging out a user is equally important. Clinicians would often have to hurry on, and would simply walk (or run) away from an ongoing session. In this case, automatic logout was deemed important.

3.4 Our Solution

There are three key principles in our design of a context-aware user authentication mechanism. First, it uses a physical token used for active gesturing and as the cryptographic basis for authentication. For this we use a smartcard running Java. Second, it uses a context-awareness system to verify the location of the user, and to log out the user when he leaves the computer(s) in a certain place. Third, it contains 'fall-back' mechanisms, so that if either of the two components in the system falls out, the user authentication mechanism switches to other mechanisms. If the context-awareness infrastructure is unreachable for any reason, the user is requested to enter his password when trying to log in. If the token cannot be accessed for any reason, the user is requested to enter both his username and password, as usual. Hence, in case of system failure the system is still usable without compromising the security completely. Note that here passwords could easily be replaced by something else, like a biometric identification.

Each client is equipped with a card reader and the authentication protocol is executed every time a user waves his card in front of it, since the card supports wireless communication.

The system architecture for the context-awareness infrastructure consists of the following main components:

- *Context Monitors* – A range of hardware and/or context data specific processes, which register changes in the environment. Examples of context monitors are location monitors based on monitoring RFID tags attached to items in the environment, or WLAN monitors that try to locate WLAN-based equipment. Other monitors might gather information about temperature, planned activities in users' personal calendars, or try to identify people in a room based on their voices.

- *Context Server* – The Context Server contains a simple data structure that stores information about *Entities* in the environment. Entities are basically people, places, or things, but this structure is extensible and all kinds of context data can be stored by implementing some simple interfaces.

By combining a context-awareness sub-system with a personal smartcard, we have designed and implemented a proof-of-concept of a proximity-based user authentication mechanism, which is both user-friendly and secure. One could argue that it seems like an unnecessary effort to implement a context-awareness system in an organization just to help users log in. However, on one hand we argue that providing easy login is essential to maintaining a smooth workflow in a hospital and on the other hand, we envision that a context-awareness sub-system is already in place in a future hospital for many other reasons, and we have demonstrated how such a system can help realize the vision of *proximity-based user authentication*.

Chapter 4

Future Work

While the work done so far covers quite different areas such as cryptographic protocols, security in pervasive computing, software architecture and implementation work, the overall theme of the author's work, is solving real world security problems based on technical, rather than administrative, solutions. The plans for for the next two years are going to be more focussed on fewer specific areas, but still within that overall theme.

The main research area is going to be security in pervasive computing. The following sections will describe ideas for future work.

4.1 Authentication and Privacy

Today the most representative technology for pervasive computing is RFID tags where privacy concerns are the dominating problem, because it is easy to get a unique identifier from an RFID tag, but RFID tags are not the only devices with that problem. Mobile phones and Bluetooth devices share the same problem. We would like to investigate various solutions to this privacy problem. For RFID tags it could involve coming up with authentication protocols for devices with few computational resources, but for other devices we might not be so constrained by the lack of computational resources, which opens up for the use of other kinds of cryptographic protocols for protecting privacy, which may not only be able to protect against passive listening, but also against leaking personal identifiers in a transaction involving the user. This is related to phishing attacks mentioned in section 4.3.

When an RFID tag has to decide whether or not to reveal its identity to a reader, it is only interested in authenticating the reader, but when RFID tags are used to identify some item or person, we need to identify the tag as well. Can both be solved in one protocol, or do we need different tags depending on which kind of authentication we need? Of course we can implement both on a device with lots of computing power, but could it be more cost-effective to make two different tags depending on their intended use? An extension of this (or perhaps an alternative approach) could be to use pseudonyms. For example an RFID tag could have one identity when read in a store and another at the user's home, which would allow full use of all the benefits

of an RFID tag with none of the disadvantages. Of course an important part of such a protocol would be that it can be made easy to use and to require very little key management from the user, since the users will in most cases be non-technical.

There are already some protocols out there for authenticating RFID tags. The most interesting one is a protocol called HB+ since it can be proven secure under the "Learning Parity with Noise" hardness assumption, where some of the other protocols are just bit-fiddling with no proof of security. The problem with HB+ is that it is not secure under a realistic threat model. If we can somehow fix HB+ we would have an efficient and secure authentication protocol for RFID tags. Also, looking at other human authentication protocols and try to adapt them to use on RFID tags is something we would also like to look at.

4.2 Using Humans as Secure Channels

A problem we encountered during the eu-DOMAIN project was how to securely add small devices to some infrastructure. More specific we have a gateway which is some powerful device (think a PC) at the users home and the user buys some device at the supermarket and brings it home. How can we associate that device with the gateway such that they can communicate securely and that the device can be sure that it is talking to the gateway and the other way around. There are a few properties of this scenario which makes it different from other seemingly similar scenarios.

1. The device has no a priori information about the gateway
2. The gateway has no a priori information about the device
3. The device has no keyboard or display device
4. The device might have very limited computing power compared to the gateway
5. Both gateway and device are operated by a non-technical user

Finding solutions to this problem seems like another interesting research area which we would like to look into. Some work has been done in that area regarding sensor networks, but this scenario is not about ad-hoc communication and we don't have the same amount of devices, so we cannot directly use their results. Most likely we will need to use humans as some kind of secure channel.

One idea could be that the user reads some code from the device and enters it on the gateway. A nice property would be that the user should not remember a full 128-bit key, but something smaller, like a PIN, and yet after the channel is set up, the security should no longer rely on the PIN being kept secret. Serge Vaudenay proposes a protocols with this property in [48] which require exponentiations of large numbers and uses four rounds of communication. It would be interesting to see if it is possible to come up with something that solves the same problem in a more efficient way. Not only designing the protocol, but considering the whole scenario including a realistic threat model (for example Vaudenay allows delivery on the communication channel to be maliciously stalled, cancelled, or replayed, which might be an unnecessary requirement if the channel is a human) and questions like what should be stored on the

device from the manufacturer, what are the minimal properties the device needs, etc. We also have hardware devices with an 8 MHz CPU and 10K RAM to implement and test this on.

4.3 Secure Operations on Insecure Devices

This area is a question of performing secure computation on a insecure device using some trusted code. Assume that we have a small hardware device with some secret information and trusted code, and we want to plug it into an computer we don't trust and perform some operation using the trusted device. As a motivating example, think of a device holding the secret key to the user's digital signature, but the document to be signed is located and displayed on an untrusted computer. The hardware device can sign the document, but the problem is how to ensure that what the computer displays on the screen and later submits to some third party is actually the document that the user thinks he signed. This looks like an authentication problem where we need to authenticate the hardware device towards the user over an insecure channel, but with an interesting twist: It needs to be verifiable by a human being. Solving this problem can lead to interesting solutions to real-world problems. First of all this will promote the use of pervasive computing as users will be able to securely sign documents using any computer when they travel. This can also be used as secure login to homebanking systems where the system is not compromised if some malicious code is running on the user's computer.

Still it does not directly solve a common problem with authentication these days, namely phishing, where a user is tricked to revealing personal information to a malicious third party, because he thinks he is talking to someone else. Currently most solutions to phishing attacks try to put stronger requirements on authenticating the user such as two-factor authentication, which doesn't really solve the problem. Authenticating the recipient of the user's personal information works better, but the best idea would be to simply not reveal any information that an attacker can use to perform the identity theft. This might or might not be a problem solveable by having the before-mentioned hardware device, but it is worth looking into how such a device could also be used to also solve problems related to phishing and identity theft. A significant portion of the research here will be figuring out the minimum requirements for the device, as well as how to do the authentication of the device towards a human being.

Bibliography

[1] C. Alberts and A. Dorofee. *Managing Information Security Risks: The OCTAVE Approach*. Addison Wesley Professional, 2002.

[2] I. Altman. *Environment and Social Behaviour*. Brooks-Cole, 1975.

[3] Anderson, Ross, Haowen, Chan, Perrig, and Adrian. Key infection: Smart trust for smart dust, 2001.

[4] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 255–270, London, UK, 2000. Springer-Verlag.

[5] P. B., A. K., and N. B. Keeping ubiquitous computing to yourself: a practical model for user control of privacy. *International Journal of Human-Computer Studies*, page In Press, 2005.

[6] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks, Feb. 2002. In Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California.

[7] J. Bardram, R. E. Kjær, and M. Ø. Pedersen. Context-aware user authentication - supporting proximity-based login in pervasive computing. In *Ubicomp*, pages 107–123, 2003.

[8] J. E. Bardram. The Trouble with Login – On usability and Computer Security in Pervasive Computing. Technical Report CfPC 2003–PB–50, Center for Pervasive Computing, Aarhus, Denmark, 2003. Available from http://www.pervasive.dk/publications.

[9] F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, pages 431–444, 2000.

[10] M. Boyle. A shared vocabulary for privacy. 2003.

[11] S. Brands. Untraceable off-line cash in wallet with observers (extended abstract). In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, 22–26 August 1993.

[12] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 93–118, London, UK, 2001. Springer-Verlag.

[13] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials, 2002.

[14] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, pages 56–72, 2004.

[15] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 410–424, London, UK, 1997. Springer-Verlag.

[16] Centre for Pervasive Healtcare. `http://www.pervasivehealthcare.dk/`.

[17] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[18] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 319–327, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[19] H. Cheung. Bluetooth sniper rifle. `http://www.tomsnetworking.com/Sections-article106.php`, 2005.

[20] H. B. Christensen and J. Bardram. Supporting human activities – exploring activity-centered computing. pages 107–116, 2002.

[21] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *Lecture Notes in Computer Science*, 1462:13–??, 1998.

[22] I. Damgård, K. Dupont, and M. Østergaard. Unclonable group identification. Cryptology ePrint Archive, Report 2005/170, 2005. `http://eprint.iacr.org/`.

[23] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order, 2001.

[24] Defcon. 3rd annual defcon wifi shootout contest. `http://www.wifi-shootout.com/`, 2005.

[25] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[26] S. Engberg, M. Harning, and C. Damsgaard Jensen. Zero-knowledge device authentication: Privacy & security enhanced RFID preserving business value and consumer convenience. In *Conference on Privacy, Security and Trust – PST*, New Brunswick, Canada, October 2004.

[27] eu-DOMAIN Consortium. eu-domain. `http://www.eu-domain.eu.com/`.

[28] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[29] H. Gilbert, M. Robshaw, and H. Sibert. An active attack against hb+ - a provably secure lightweight authentication protocol. Cryptology ePrint Archive, Report 2005/237, 2005. `http://eprint.iacr.org/`.

[30] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[31] K. M. Hansen, S. B. Larsen, J. I. Pagter, M. Pedersen, and J. Thomsen. An evaluation of a residential pervasive computing platform based on osgi. In *Proceedings of the IADIS Applied Computing Conference*, pages 246–253. ACM Press, 2004.

[32] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *Proceedings of the 5th Annunal ACM/IEEE Conference on Mobile Computing and Networking (MobiCom99)*, pages 59–68, Seattle, WA, USA, 1999. ACM Press.

[33] IBM Research. Mobile and pervasive computing. `http://www.research.ibm.com/compsci/spotlight/mobile/`.

[34] A. Juels. Minimalist cryptography for low-cost RFID tags. In C. Blundo and S. Cimato, editors, *International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italia, September 2004. Springer-Verlag.

[35] A. Juels, R. L. Rivest, and M. Szydlo. The blocker tag: selective blocking of rfid tags for consumer privacy. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 103–111, New York, NY, USA, 2003. ACM Press.

[36] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In V. Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3126 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. IACR, Springer-Verlag.

[37] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *EUROCRYPT*, pages 198–214, 2005.

[38] S. Lederer, J. Mankoff, and A. K. Dey. Towards a deconstruction of the privacy space. 2003.

[39] A. Lysyanskaya. Signature schemes and applications to cryptographic protocol design, 2002.

[40] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science, Kingston, Canada, August 2005. Springer-Verlag.

[41] L. Palen and P. Dourish. Unpacking "privacy" for a networked world. In *Proceedings of the conference on Human factors in computing systems*, pages 129–136. ACM Press, 2003.

[42] M. Ø. Pedersen, J. I. Pagter, and T. P. Pedersen. Pervasive computing: IT security and privacy. 2004.

[43] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.

[44] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, New York, NY, USA, 1990. ACM Press.

[45] SECURE. Secure: Secure environments for collaboration among ubiquitous roaming entities. `http://www.dsg.cs.tcd.ie/dynamic/?category_id=-30`.

[46] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for adhoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, London, UK, 2000. Springer-Verlag.

[47] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2 edition, 2001.

[48] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *CRYPTO*, pages 309–326, 2005.