KUOPION YLIOPISTON JULKAISUJA H. INFORMAATIOTEKNOLOGIA JA KAUPPATIETEET 10
KUOPIO UNIVERSITY PUBLICATIONS H. BUSINESS AND INFORMATION TECHNOLOGY 10

MARKO HASSINEN

Studies in Mobile Security

Doctoral dissertation

To be presented by permission of the Faculty of Business and Information Technology of the University of Kuopio for public examination in Auditorium L2, Canthia building, University of Kuopio, on Friday 16th November 2007, at 12 noon

Department of Computer Science
University of Kuopio

KUOPION YLIOPISTO

KUOPIO 2007

Distributor:	Kuopio University Library P.O. Box 1627 FI-70211 KUOPIO FINLAND Tel. +358 17 163 430 Fax +358 17 163 410 www.uku.fi/kirjasto/julkaisutoiminta/julkmyyn.html
Series Editors:	Professor Markku Nihtilä, D.Sc. Department of Mathematics and Statistics Assistant Professor Mika Pasanen, Ph.D. Department of Business and Management
Author's address:	Department of Computer Science University of Kuopio P.O. Box 1627 FI-70211 KUOPIO FINLAND Tel. +358 17 162 559 Fax +358 17 162 595 E-mail: Marko.Hassinen@uku.fi
Supervisors:	Professor Martti Penttonen, Ph.D. Department of Computer Science University of Kuopio Lecturer Ville Leppänen, Ph.D. Department of Information Technology University of Turku
Reviewers:	Professor Matti Linna, Ph.D. Department of Computer Science University of Vaasa Professor Jukka Manner, Ph.D. Telecommunications Software and Multimedia Laboratory Helsinki University of Technology
Opponent:	Valtteri Niemi, Ph.D. Nokia Research Center

ISBN 978-951-781-989-3
ISBN 978-951-27-0108-7 (PDF)
ISSN 1459-7586

Kopijyvä
Kuopio 2007
Finland

Hassinen, Marko. Studies in Mobile Security. Kuopio University Publications H. Business and Information Technology 10. 2007. 58 p.
ISBN 978-951-781-989-3
ISBN 978-951-27-0108-7 (PDF)
ISSN 1459-7586

ABSTRACT

The security of mobile communication has become increasingly important with the development of devices that feature more and more versatile communication functionality. The advent of user programmable mobile phones has made it possible to create third party software that needs security beyond what the network can provide. Third party software in mobile phones has expanded into areas where the user expects security in forms of confidentiality and integrity of data. At the same time this development has lead into a situation where mobile communication is used in areas such as mobile commerce and mobile payments. Providers of these kinds of services expect to have security in the form of strong authentication and non-repudiable transactions.

Both providers and users have their expectations for the security of mobile services. This thesis discusses these requirements and provides solutions that fulfil the expectations. Security is required in communication as well as in the storage of data that is generated as a result of this communication. Software solutions presented in this thesis utilise both symmetric and asymmetric ciphers. Symmetric encryption is used to provide confidentiality of message exchange and authentication, whereas asymmetric encryption is used for purposes of non-repudiation. Integrity of data is obtained using hash functions combined with encryption.

Mobile commerce and mobile payment require non-repudiation so that none of the parties can later deny their participation in transactions they have committed to. Non-repudiation is obtained by asymmetric encryption using a public key infrastructure (PKI). The assumption that the private key of the user is possessed only by that user means that nobody else can create a valid digital signature of that user. In our work we have used a governmentally administered PKI to implement protocols for mobile payments. With these protocols secure mobile payments can be achieved both in real and virtual point-of-sale (POS). Using a governmental PKI the protocol authenticates citizens instead of customers of a certain financial institution. This means that the system is open to any merchant, service provider and financial institution.

Secure storage of exchanged messages is important in the case where the device is accessed by an unauthorised party. This may happen for example if the device is lost or stolen. The software solutions presented in this thesis allow access to stored data only to a user with proper credentials.

Security in mobile communication can be achieved with current technology and equipment. This can, furthermore, be done in a way that can benefit a large audience. The security level can be set high enough to provide secure financial transactions using a mobile phone.

Universal Decimal Classification: 004.056, 003.26, 621.395.721.5
Inspec Thesaurus: security; telecommunication security; security of data; mobile communication; mobile handsets; electronic commerce; cryptography; cryptographic protocols; public key cryptography; Java; message authentication; data integrity; secure storage

Acknowledgments

This thesis is the result of research carried out at the Department of Computer Science, University of Kuopio.

I would like to thank professor Martti Penttonen from the University of Kuopio for his guidance and advice in writing this thesis and the articles it is based on. Also I would like to thank professor Pekka Kilpeläinen for his valuable advice with the articles I have written. Dr. Mauno Rönkkö deserves a special expression of gratitude for reading a number of our manuscripts and giving detailed advice. In preparing this summary Dr. Paula Leinonen gave me a helping hand in the technical layout work saving me a lot of time. Thank you.

Professor Smile Markovski from the Institute of informatics, University of Skopje invited me to Macedonia to do research work, taught me a numerous things about it and also introduced the question of encrypting SMS messages that later became the core substance of this thesis. Smile, Blagodaram mnogu. The biggest gratitude I must address to my co-authors, without whom all this work would have been much less fun and might not ever been concluded. Konstantin Hyppönen not only co-authored some of the most important papers, but also helped me with the summary in many ways.

And last, but not least, a special thank you to all my colleagues at the Department of Computer Science for numerous helpful talks and discussions in corridors and at lunch tables.

Abbreviations and notations

Abbreviation	Description
3G	3rd Generation Mobile Network
APDU	Application Protocol Data Unit
API	Application Programming Interface
BTS	Base Transceiver Station
CA	Certificate Authority
CLDC	Connected Limited Device Configuration
CRL	Certificate Revocation List
EMS	Emergency Medical Services
EMV	Europay-Mastercard-Visa
FINEID	Finnish Electronic Identity
FINUID	Finnish Electronic Identity User Identification
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
FSP	Financial Service Provider
HLR	Home Location Register
J2ME	Java 2 Micro Edition
JAD	Java Application Descriptor
JAR	Java Archive
JCE	Java Cryptography Extension
MIDP	Mobile Information Device Profile
MIME	Multipurpose Internet Mail Extension
MS	Mobile Station (Mobile user)
MSC	Mobile services Switching Centre
NFC	Near Field Communication)
OCSP	Open Certificate Status Protocol
OTA	Over The Air provisioning
PKCS12	Public Key Cryptography Standard #12
PKI	Public Key Infrastructure
POS	Point-of-Sale
PRC	Population Register Centre
PSP	Payment Service Provider
RA	Registration Authority
RFID	Radio Frequency Identification
RSA	Rivest Shamir Adleman
SAT	SIM Application Toolkit
SATSA	Security and Trust Services API
SIM	Subscriber Identity Module
SMS	Short Message Services
TBS	To Be Signed
TTP	Trusted Third Party
URL	Universal Resource Locator
USB	Universal Serial Bus

SOAP	Simple Object Access Protocol
VLR	Visitor Location Register
WAP	Wireless Application Protocol
WIM	WAP Identity Module
WMA	Wireless Messaging API
WSDL	Web Service Definition Language
WTK	Wireless Toolkit
WTLS	Wireless Transport Layer Security
XML	Extensible Markup Language

Notation	**Description**
K_i	A shared secret key between a mobile user and the home network
K_c	A shared session key between a mobile user and the home network

List of the original publications

This thesis is based on the following published articles, which are referred to in the text by their Roman numerals:

I Hassinen M, Markovski S.: Secure SMS messaging using quasigroup encryption and Java SMS API. *Proceedings of the Eighth Symposium on Programming Languages and Software Tools.* Kuopio, Finland. University of Kuopio 2004.

II Hassinen M., Markovski S.: Differential Cryptanalysis of the Quasigroup cipher. *Proceedings of the Finnish Data Processing Week.* Petrozavodsk, Russia. Petrozavodsk State University 2004.

III Hassinen M.: SafeSMS-End-to-end encryption for SMS messages. *Proceedings of the 8th International Conference on Telecommunications.* Pages:359 - 365. Zagreb, Croatia. IEEE 2005.

IV Hassinen M., Hyppönen K.: Strong Mobile Authentication. *Proceedings of the 2nd International Symposium on Wireless Communication Systems.* Pages:96 - 100. Siena, Italy. IEEE 2005.

V Hassinen M., Laitinen P.: Secure SMS messaging in the Health Care Domain. *Proceedings of the 19th International Congress of the European Federation for Medical Informatics.* Geneva, Switzerland. ENMI 2005.

VI Hassinen M., Marttila-Kontio M., Saesmaa M., Tervo H.: Secure two-way transfer of measurement data. *International Journal of Network Security. Vol. 6, No. 1, 2008, pp. 53-59.* Taichung, Taiwan. IJNS 2007.

VII Hassinen M.:Java-based Public Key Infrastructure for SMS messaging. *Proceedings of the 2nd IEEE International Conference on Information & Communication Technologies: from Theory to Applications - ICTTA'06.* Damascus, Syria. IEEE 2006.

VIII Hassinen M., Hyppönen K., Haataja K.: An Open, PKI-Based Mobile Payment System. *Proceedings of the International Conference on Emerging Trends in Information and Communication Security - ETRICS 06.* Freiburg, Germany. Springer LNCS 2006.

IX Hassinen M., Hyppönen K., Trichina E.: Utilizing national public-key infrastucture in mobile payment systems. *Electronic Commerce Research and Applications* (2007), in press. doi: 10.1016/j.elerap.2007.03.006

Author’s Contribution

This dissertation is the result of research carried out at the Department of Computer Science, University of Kuopio and at the Institute of Informatics, University of Ss. Cyril and Methodius, Skopje, Republic of Macedonia.

Publication I represents an application for end-to-end encryption of SMS messages using the Quasigroup cipher. The software implementation and its analysis were done by the author, the theoretical background was done jointly by the author and Professor Smile Markovski.

Publication II contains results of differential analysis of the quasigroup cipher. The original idea, formulation of the method, and its implementation were done by the author.

Publication III represents a secret key implementation for SMS messaging and delivers confidentiality, authentication, and integrity of SMS messages. The application was solely implemented by the author.

Publication IV introduces the idea of using a national public key infrastructure in messaging and mobile payments. The original payment protocol introduced in this publication was designed by the author and the implementation on the mobile device came from the author. Smart card implementation and simulation software came from Konstantin Hyppönen. The article was jointly authored.

Publication V shows that secure SMS messaging can be utilised in health care where need for security is obvious. The technical part was authored by Marko Hassinen and the health care specific part was authored jointly. The customisations of the application were done by the author.

Publication VI introduces a secure two-way transfer of measurement data where the security part was implemented and authored by the author. The work was originally published in the Proceedings of the Third International Conference on Information Technology : New Generations and selected from the conference proceedings to the International Journal of Network Security.

Publication VII was solely written by the author. The article describes an application that uses public-key cryptography to provide confidentiality, authentication, integrity, and non-repudiation. The design and implementation were done by the author.

Publication VIII refines the mobile payment system and the protocols it is based on. The author wrote the implementation details of the paper and the protocols were jointly refined with Konstantin Hyppönen. Most of the article was co-authored by Hassinen and Hyppönen, Haataja wrote the part about Bluetooth and NFC.

Publication IX represents a wider view of mobile payment in general and a detailed description of the payment system developed by the authors. The article went through several revisions, during which all the authors co-authored most of the text. The protocols were already defined earlier, the proof-of-concept implementation and its details came from Marko Hassinen.

Contents

Chapter 1

Introduction

Mobile communications have progressed in a very fast pace. As communication speeds in mobile environments have increased, developing mobile applications and communication solutions has become feasible. This has also created new challenges in areas like security, especially in mobile networks, where being anonymous is easier than before. Solving security issues, such as strong authentication, confidentiality or privacy and integrity of data are challenges that have to be considered. Without adequate security, several areas of applications can not utilise the mobile technology to its fullest potential.

SMS messages are sometimes used for the interchange of confidential data such as social security number, bank account number, password etc. A typing error in selecting a number when sending such a message can have severe consequences if the message is readable to any non-intended receiver. Most mobile operators encrypt all mobile communication data, including SMS messages but sometimes this is not the case, and even when encrypted, the data is readable for the operator. Among others these needs give rise for the need to develop additional encryption for SMS messages, so that only accredited parties are able to engage communication. Sometimes SMS messages are used in mobile commerce, such as buying entrance tickets [30], in which case they may convey considerable value.

1.1 Security structures of the GSM network

Authentication in the standard GSM network is based on a secret key K_i that is shared by the mobile device and the home operator. On the mobile device this key is stored in the SIM card. Each mobile user is assigned an IMSI (International Mobile Subscriber Identity), which is used to identify the user for billing purposes. Since there is a need to protect the identity of a user, a TMSI (Temporary Mobile Subscriber Identity) is assigned to the user in the authentication procedure and used instead of the IMSI [28].

In the authentication process the MS (Mobile Station) contacts the BTS (Base Transceiver Station) which in turn contacts the MSC (Mobile Switching Centre). If the MS is known in the MSC, the TMSI is used as an identity to the MS and the MSC probably already has adequate information to authenticate the MS [49]. In case the MSC has no prior knowledge of the MS, it will ask for the IMSI of the user. Having the IMSI, the MSC will contact the home network of MS to get the authentication information. The HLR (Home Location Register) of the home network of MS will then generate one or more (often five in practise) triplets each containing a random challenge (RAND) a signed response (SRES) to the challenge RAND and a session key K_c. SRES and K_c are based on the shared key K_i [28].

Once the MSC has the triplets, it can use them for authenticating the MS. The MSC sends the MS one of the triplets as a challenge. Since the MS knows K_i it can derive the SRES as well as the K_c. After the MS has responded with the SRES, the MSC compares the SRES obtained from the HLR with the one received from the MS and if they match, concludes that the MS's claimed identity is authentic. Now the MS and MSC share the key K_c, which is further used to encrypt the traffic between the MS and the MSC.

Although somewhat out of the scope of this thesis, there is an attack using a false base station to find out the IMSI identity of the mobile user. Solutions we present do not protect the user from this type of an attack and it seems that an obvious solution to this problem in the current GSM system is not available. The algorithms used to generate the SRES and encrypt the air-to-air traffic were originally kept secret. Later on some of these algorithms leaked out to the public and have been analysed by the scientific community. Several weaknesses have been found in these algorithms, giving opportunities to attack the system. The GSM encryption can be broken in less than one second with only few dozen milliseconds of off-the-air conversation [2]. However, as the next chapter will demonstrate, weaknesses in these algorithms and protecting the user from them is not the main point of this thesis. We will show that, despite the shortcomings in the security of the GSM network, a scheme can be created which provides security, integrity, non-repudiation and confidentiality of traffic in SMS messaging and packet data.

1.2 Built-in security of SMS messaging in the GSM network

The GSM network only encrypts the air-to-air traffic. The traffic on the operators network is in plaintext [33]. The encryption between the mobile device and the BTS protects the data from an eavesdropper that listens to the air traffic. However, there is no security mechanism protecting the data inside the operators network. This means that the operator can at will listen to any voice communications and read any SMS messages. Furthermore, the security system does not provide any kind of non-repudiation. This means that a mobile user can repudiate any message. Since messages travel in plain text, the network operator can manufacture any message claiming that it was sent by the mobile user.

1.3 Desired security features of SMS messaging

As stated before, SMS is a widely used service that is well adopted by the large public. At the same time SMS technology is available world-wide and, hence, can be utilised for numerous purposes. A limiting factor, however, in applications such as electronic payments, has been the lack of security (see VIII for a survey). We will show that the following security principles can be implemented using SMS messaging.

Authentication means that when a communicating party X claims to be X, we can either be convinced that it is true or we can discover that X is in fact someone else and hence lying.

Non-repudiation means that we can create commitments that are undeniable at a later date. If party X commits to having produced a message M then later on this commitment can be verified, and X can not deny having committed to M.

Confidentiality means that delicate information is available only to those parties that it is meant for. An eavesdropper can not find out the content of confidential messages without adequate credentials, such as an encryption key.

Integrity of data means that data can travel safely without a threat of being manipulated, changed on the way. If data is manipulated, or tampered with, we can detect it. We can not in general repair the tampered data, but it has to be retransmitted.

Furthermore, we will demonstrate applications of these security features in areas of mobile commerce (publications VIII and IX), health care (publication V), and in a measurement system (publication VI).

Chapter 2

Theoretical foundations

In order to be able to deliver the security features described in the previous chapter we need to have cryptographic procedures. In the first application (publication I) that was intended to fulfil some of these requirements an encryption method called Quasigroup encryption was used. The Quasigroup encryption is especially well suited for mobile devices due to the simplicity of the encryption/decryption procedure. The encryption is essentially a sequence of table lookups and there are no computationally complex arithmetic operations such as modular exponentiation. This makes the procedure very efficient on devices with limited processing capabilities. Since the operations are very simple, there is no need for specialised hardware for mathematical operations.

The following section describes this encryption method as well as some results of the cryptanalysis of this cipher. The Quasigroup method has not been studied very much from the cryptanalytic perspective in literature. The cryptanalysis we made utilises the technique of differential cryptanalysis, which has become a common tool in cryptanalysis. Our research (Publication II), however, uses a modified differential method, since the original method was designed for an iterated block cipher which the Quasigroup cipher is not.

With the quasigroup method confidentiality of data and one-factor authentication can be achieved, but for the other requirements additional measures are required. Hash functions combined with a PKI (Public Key Infrastructure) can be used to provide integrity, non-repudiation and strong two-factor authentication. Futher sections handle the PKI infrastructure that has been used to provide these security functions.

2.1 Quasigroup encryption

In order to introduce the Quasigroup encryption method, we here give some necessary definitions. A *groupoid* is a finite set Q that is closed with respect to an operator $*$, i.e., $a * b \in Q$ for all $a, b \in Q$. A groupoid is a *quasigroup,* if it has unique left and right inverses, i.e., for any $u, v \in Q$ there exist unique x and y such that $x * u = v$ and $u * y = v$ [25].

This means that all operations are invertible and have unique solutions, which implies their usability as cryptographic substitution operations. With this in mind we can define inverse operations for $*$, denote them $\backslash$ (left inverse) and / (right inverse) . The operator $\backslash$ (resp. /) defines a new quasigroup $(Q, \backslash\ )$ (resp. $(Q,\ \ )$) and for algebra $(Q, \backslash, *)$

$$x * (x\backslash y) = y = x\backslash(x * y) \tag{2.1}$$

A quasigroup can be characterised with a structure called Latin square. A $Latin$ $square$ is an $n{*}n$ matrix where each row and each column is a permutation of elements of a set. In our case $|Q| = n$.

Several other operations can be derived from the operation $*$ [26], but for our purposes operations $*$ and $\backslash$(right inverse) are sufficient.

2.1.1 Encryption

The encryption primitive $e_l, l \in Q$ on sequence $x_1x_2 \dots x_n$ is defined as
$e_l(x_1x_2 \dots x_n) = y_1y_2 \dots y_n$ where

$$\begin{cases} y_1 = l * x_1, \\ y_{i+1} = y_i * x_{i+1}(i = 1, \dots n-1) \end{cases} \tag{2.2}$$

The element l is called the $leader$ and in our application it is derived from the secret key (password). Transformation e_l is a mapping $A^+ \rightarrow A^+$. Elements $x_i, y_i \in A$ are usually not characters, but entities of bits, where the bitlength of the element is defined by the size of the Latin square associated with the quasigroup.

2.1.2 Decryption

Decryption $d_l : A^+ \rightarrow A^+$ is naturally a reverse operation of the encryption. It is defined as $d_l(y_1y_2 \dots y_n) = x_1x_2 \dots x_n$, where

$$\begin{cases} x_1 = l\backslash y_1, \\ x_{i+1} = y_i\backslash y_{i+1}(i = 1, \dots n-1) \end{cases} \tag{2.3}$$

In essence, the decryption primitive as well as the encryption primitive are table lookup methods, where the Latin square associated with the quasigroup is used. In encryption the previous encrypted element and the next not yet encrypted element are used to find the corresponding value from the table. This value is then used as the new encrypted value.

In decryption the values used for lookup are the previous already decrypted value and the next not yet decrypted value. With these a new decrypted value is found from the Latin square.

One application of the encryption method does not provide adequate security, since cryptanalysing that would be a trivial task. For this reason a combination of encryption and decryption primitives can be used with multiple quasigroups. In Fig. 2.1.2 is a structural view of the encryption process where $a_1a_2 \dots a_n$ represents the plaintext. $b_1b_2 \dots b_n$ represents the ciphertext after one application of encryption and $c_1c_2 \dots c_n$ after two applications.

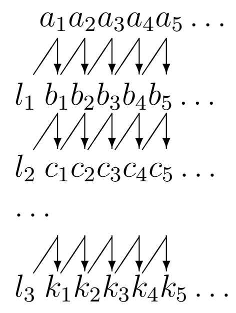
$a_1a_2a_3a_4a_5\dots$
l_1 $b_1b_2b_3b_4b_5\dots$
l_2 $c_1c_2c_3c_4c_5\dots$
$\dots$
l_3 $k_1k_2k_3k_4k_5\dots$

Figure 2.1: Structural view of encryption with k-1 iterations of the cipher

2.1.3 Differential Cryptanalysis of the Quasigroup Cipher

Differential cryptanalysis was originally presented by Eli Biham and Adi Shamir, two Israeli cryptographers. Their analysis of DES (Data Encryption Standard) using differential cryptanalysis showed how some bits of the key can be discovered in block ciphers that have a Feistel structure [3].

The method of differential cryptanalysis is based on bitwise differences (XORs) of input blocks and the analysis of corresponding output XORs of a certain component of the cipher. These differences are collected into a table called a *difference distribution table*. In the original paper by Biham and Shamir the target of this analysis was the S-boxes of DES [4].

Differential attack is a chosen plaintext attack that uses a large amount of input block pairs which have some predetermined bitwise difference. The goal of the attack is to discover bits of a subkey used in a round of the cipher.

Differential cryptanalysis of the Quasigroup cipher revealed that one should use Latin squares of higher order than four (see II for details). Considering a practical implementation of the cipher this means that order eight would be the smallest Latin square that gives adequate security and is practical to implement. In general, quasigroups of order 2^k, where k is a small integer, are suitable for practical implementations. This is due to the fact that with an order four quasigroup one can encrypt two bits, with order eight three bits and so on.

The complexity of the attack based on differential cryptanalysis of a Quasigroup cipher depends on several variables. These include the amount of encryptions, the number of used quasigroups, the order in which they have been applied and whether any of these details are known to the attacker. The main idea of differential cryptanalysis on the quasigroup cipher is to discover the structure of the Latin square used in the encryption and ultimately the values of each entry in that Latin square. Finding the structure of the quasigroup would give clear advantage compared to exhaustive search.

We derived a difference distributions table for a certain quasigroup using Algorithm 1, where $\oplus$ denotes a bitwise XOR operation.

```
createDifferenceTable(Quasigroup Q):difference distribution table
(1)    for ($a_1$ := 0 . . . $|Q| - 1$)
(2)      for ($a_2$ := 0 . . . $|Q| - 1$)
(3)        for (leader := 0 . . . $|Q| - 1$)
(4)          $c_1$ := e_transformation(leader,$a_1$)
(5)          $c_2$ := e_transformation(leader,$a_2$)
(6)          input_xor := $a_1 \oplus a_2$
(7)          output_xor := $c_1 \oplus c_2$
(8)          distributions[input_xor][output_xor]++
(9)        endfor
(10)     endfor
(11)   endfor
```

Algorithm 1. Algorithm used to generate pairs distribution tables from a quasigroup.

This analysis showed that there are some Latin squares of order 4 that produce quasigroups that are weak against differential cryptanalysis (see II for details). However, on larger Latin squares weakness against differential cryptanalysis was found only on very symmetric Latin squares. Analysis against non symmetric Latin squares of order larger than 4 was not successful.

We also carried out a differential cryptanalysis to find a structure of the Latin square from a ciphertext-plaintext pair. This analysis is statistical and requires quite a lot of plaintext and ciphertext. The main goal of this analysis is to find the structure of the Latin square. After the structure is found, the actual quasigroup can be found using a relatively easy exhaustive search (see II for details).

To analyse the cipher we created an algorithm (Algorithm 2) for generating statistical pairs distribution tables from ciphertext-plaintext pairs with multiple encryptions. In the algorithm a_i refers to the i:th element of plaintext. Respectively c_i refers to the i:th element of the ciphertext.

```
createDifferenceTable(plaintext $a_1 \dots a_n$, ciphertext $c_1 \dots c_n$):difference distribution table
(1)    assign $leaders[quasigroup\ order]$ with zeros
(2)    while values left in ciphertext
(3)      take the next n values $c_i \dots c_{i+n}$
(4)      if leaders[$c_i$] = 0
(5)        leaders[$c_i$] := 1
(6)        foreach ($c_k \dots c_{k+n}, c_i \dots c_{i+n} = c_k \dots c_{k+n}, i \neq k$)
(7)          input_xor := $a_{i+n+1} \oplus a_{k+n+1}$
(8)          output_xor := $c_{i+n+1} \oplus c_{k+n+1}$
(9)          distributions[input_xor][output_xor]++
(10)       endforeach
(11)     endif
(12)   endwhile
```

Algorithm 2. Algorithm used to generate statistical pairs distribution tables from a ciphertext-plaintext pair (n encryptions).

Although successful with Latin squares of small oder, the statistical analysis required a very large amount of ciphertext-plaintext pairs for quasigroups of larger order with multiple encryptions and did not suceed in most cases.

2.1.4 Generating Latin Squares

To generate a Latin square of order n, one needs an algorithm. A theory called "System of different representatives" [27] has been used to generate random Latin squares. It is also possible to assign values to a Latin square randomly, but this seems to be a slow process in which a lot of tries has to be done. Since in the Latin square each row and column must be a permutation of the values of the set, one has to back up very often in the process of assigning the values in oder to maintain this requirement.

Finding a system to map another credential, for example a password, into a Latin square would be a breakthrough for the quasigroup encryption. As Latin squares are currently used as shared secrets, one needs to have a mechanism to distribute these keys securely. A downside to all this is that a Latin square of cryptographically safe order easily takes more space that encryption keys of comparable safety level in other ciphers. This, however, can not be generalised as there is a trade-off in the quasigroup encryption in terms of key size and encryption speed. Using smaller Latin squares means that more iterations of the encryption needs to be done. The major advantage when using a large Latin square is very easy and fast encryption. Using a 256*256 Latin square means that encryption can be done a byte at a time resulting a fast encryption with keysize of 64 kB. The number of Latin squares of this size is huge, the exact number is not known (see II, p. 12. for details).

2.2 PKI and the FINEID infrastructure

Symmetric encryption can provide confidentiality and, combined with cryptographic hash functions, integrity of data. It can also be used for authentication assuming that the shared secret is known only by the communicating parties. However, applications such as mobile payment solutions require non-repudiation which can not be achieved using symmetric encryption.

Public key encryption has become probably the most important security concept of computer communications since it was first introduced by Diffie and Hellman in 1976. We have used public key encryption to achieve the security requirements stated earlier for SMS messaging. Two different public key infrastructures (PKI's) have been in use in our research. One has been based on a key pair created on the device of the user (in publication VII) and the other one has been based on a PKI provided by the Finnish government.

To realise the PKI in publications IV, VII, and IX, we used the PKI provided by the Finnish Population Register Centre [34]. The centre issues electronic identity cards to Finnish citizens and permanent residents. The card holder's private keys are stored in the memory of this tamper resistant card. Moreover, there are no other copies of these keys, and it is practically impossible to manufacture duplicates of the card. This suits perfectly our requirements for authentication and non-repudiation.

Finnish Electronic Identification (FINEID) [35] application manages the contents of the electronic identity card and provides a command interface for performing private key operations, such as, creating a digital signature. User authentication is accomplished using PIN (Personal Identification Number) codes.

Population Register Centre maintains an online certificate directory, namely the FINEID-directory, which can be used to retrieve certificates of other users. Public keys of each registered individual can be downloaded upon a search with appropriate criteria. Besides that, a CRL (Certificate Revocation List) of invalid certificates is available from the FINEID-directory. A certificate is placed on this list, for example, in case the card containing the corresponding private key is lost.

Chapter 3

Technological foundations

In order to create an application that fulfils the security requirements we stated earlier, several technological aspects have to be considered. In this chapter we give the necessary technological background to understand how the applications were implemented and what affected the decisions we made with the implementation. A reader familiar with the Java Micro Edition (J2ME) may want to skip the chapter and refer to it as needed.

As our solutions are Java applications, we cover the J2ME environment and some of its APIs (Application Programming Interface) which are important for our purposes.

3.1 J2ME

J2ME (Java 2 Micro Edition) is a runtime environment specifically designed for devices with very limited resources such as mobile phones or handheld computers. J2ME is comprised of CLDC (Connected Limited Device Configuration) and Mobile Information Device Profile (MIDP). A program developed for J2ME is called a MIDlet. MIDlets use classes defined in APIs of CLDC and MIDP. There is no straight interaction between a MIDlet and the device itself, since MIDlets are run by the Java virtual machine (JVM). The architecture of J2ME is depicted in Figure 3.1. This architecture limits the functionality a MIDlet can have into those provided by the runtime environment.

Basic Java language features and language libraries are contained in the CLDC (Connected Limited Device Configuration). It contains API classes in the very heart of the Java language, such as basic language constructs, input/output functionality, utilites, and networking. Built on top of the CLDC, see Figure 3.1, the MIDP provides classes that give additional functionality for a specific type of device [18]. The OEM (Original Equipment Manufacturer) specific classes are classes provided by the device manufacturer and are not present in all MIDP devices.

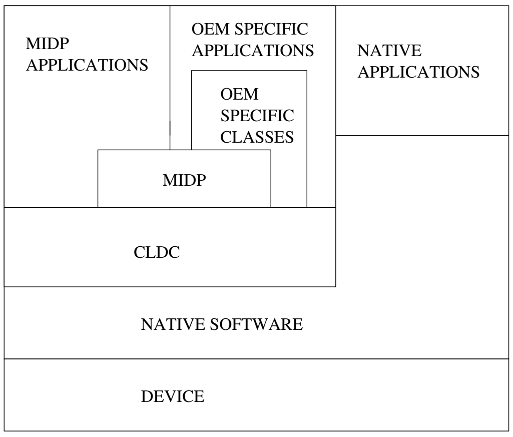

Figure 3.1: J2ME architectural structure [18].

3.1.1 Wireless messaging API - WMA

The WMA is described in specification JSR-120 [19]. This API provides the means for sending and receiving SMS messages (among other functions) using the GCF (Generic Connection Framework), which is a part of the CLDC. CGF is also used by other types of connections, such as TCP/IP and IrDA [40].

In the WMA, the class *javax.microedition.io.Connector* is used for creating a connection. A new connection for SMS traffic is created as a *MessageConnection* object:

```
MessageConnection clientConn =
  (MessageConnection)Connector.open
  (protocol + outgoingaddress);
```

For SMS messages the protocol is `sms://`. The outgoing address is of the form `number:port`, for example `+5550000:4321`.

The *javax.wireless.messaging.MessageConnection* class has methods for sending and receiving SMS messages. Sending a message (byte array *toSend*) can be done as described below:

```
String sAddr = "sms://+5550000:4321";
MessageConnection clientConn =
 (MessageConnection)Connector.open(sAddr);
BinaryMessage message =
 (BinaryMessage)clientConn.newMessage(
  MessageConnection.BINARY_MESSAGE, sAddr);
message.setAddress(sAddr);
message.setPayloadData(toSend);
clientConn.send(message);
```

When receiving an SMS message, the application has to open a listening connection for incoming messages. In our application (publication I) we implemented a method named *openListeningSMSConn*, which opens a connection and sets a listener to listen to incoming messages.

```
protected boolean openListeningSMSConn()
{
  try
  {
    conn = (MessageConnection) Connector.open("sms://:"+sPort);
    conn.setMessageListener(this);
  }
  catch(Exception e)
  {
     conn = null;
  }
  return (conn != null);
}
```

The protocol always contains the string `sms://`, and the variable *sPort* is defined by the application.

Once the connection is open, an instance of a message listener class implementing the *MessageListener* interface is set using the *setMessageListener* method of the Connection. A class implementing the *MessageListener* has to implement the *notifyIncoming* method, which the application wakes up upon receiving a new message.

3.1.2 Security and Trust Services API - SATSA

SATSA (Security and Trust Services API) is defined in the JSR-177 [17]. Among other things it gives the programmer tools for handling the smart card properties of the phone's SIM card. In order to use the SIM as a cryptographic processor, there has to be a way to communicate with the card. Using SATSA it is possible to send commands to the card and get responses. These commands are called APDUs (Application Protocol Data Unit). An example of this would be to send the card a message which the card would sign using the private key stored on the card. The card would then return the signature and the private key would never leave the card.

WIM (Wireless Identity Module) is a tamper resistant device, e.g. a smart card, intended to be used in WTLS (Wireless Transport Layer Security) for identification and authentication purposes [52]. In a mobile device WIM can be located on the SIM card or on an external smart card [32]. WIM on a smart card can be accessed using APDUs and the information structure follow the PKCS15 [37] standard.

3.1.3 Implementing permanent storage with Record Store

The Record Store (*javax.microedition.rms*) is a package containing classes used for permanent storage. Details of how the storage is implemented on the device are left to the manufacturer, and the API provides the programmer means to use the storage without paying attention to details, such as file reading and writing. The device is also responsible for guaranteeing that information in the storage is kept safe during power off's [18].

A storage is opened with *openRecordStore* method. Records in the store can be iterated using the *enumerateRecords* method and a new record can be inserted with the *addRecord* method.

Methods for adding and retrieving records use byte arrays, which means that an object to be stored must first be represented as an array of bytes. Similarly, the method for retrieving data will return a byte array, which has to be converted into an object. As neither MIPD nor CLDC implements java.io.Serializable [42], in our applications each object that is stored, has a method for making a byte array presentation of the objects state. Respectively, each stored object has a constructor that accepts a byte array as a parameter and constructs the object with its original state.

In our messaging application presented in publication III, sent and received messages are stored using the record store. Similarly, for easier use of the application, we implemented a phone book with contacts. We used a personal user password to derive an encryption key to encrypt certain values of stored messages as well as contacts. For contacts, all values, hence name, number, and password are encrypted. This way the contacts can be read only with the correct user password. Also different users can use the same phone book and will only see their own contacts.

The payload of a stored message is already encrypted during transmission. With sent messages also the date and the number of the recipient is encrypted, so that a user sees only messages he has sent. For incoming messages the number is not encrypted, since the user password is not always available when the message is received. This is the case when the application is not running at the time of receiving a message. After reading a received message, the user may choose to encrypt the message details, after which, the message can be seen only in the inbox of that user.

Encryption of elements in the record store guarantees confidentiality even in cases where the device is stolen or lost and hence could be accessed by an adversary. According to the European Commission [8], in Netherlands alone 198000 mobile phones were stolen in 2002 and the Danish police estimated that 0,5 percent of mobile phones are stolen yearly. Amount of accidentally lost or misplaced phones is probably much greater, but reliable statistics are very difficult to find.

3.1.4 Cryptographic algorithms in J2ME

The J2ME does not specify means for doing cryptographic calculations. There are no ready made classes for cryptographic processing in either MIDP or CLDC. This means that one either has to implement them oneself or use a code from a third party. For the Quasigroup encryption (in publication I) we implemented the algorithm ourselves. For the PKI (in publication VII) we used a ready made package from the Legion of the Bouncy castle. This package includes a large variety of different cryptographic algorithms and helper classes [24].

3.1.5 Push registry

MIDP 2.0 introduced a construct called *Push registry*, which enables an application to register inbound connections to AMS (Application Management Software). This feature allows AMS to start an application on arrival of a message that is meant for the application.

Registering an application to the Push Registry can be done in two ways. We use the static method of describing the Push Registry entry in the application descriptor file. The line

```
MIDlet-Push-1: sms://:54321,SafeSMS,*
```

in the jad file registers all SMS messages with port 54321 to be handled by the SafeSMS application (see III). This means that upon arrival of a text message on port 54321, the message is directed to SafeSMS. In case the application is not running, it will be lauched by the AMS. Static registrations last in the AMS registry until the application is removed from the device.

The other way to make entries into the Push Registry is to do it at runtime inside the application. This is referred to as the dynamic approach. A connection is registered using the *registerConnection* method and can be removed with the *unregisterConnection*.

Sometimes it is necessary to know whether the application was started by the user or by the AMS using the Push registry. Method *PushRegistry.listConnections(true)* can be used to detected whether the application was started by AMS after an incoming connection [43]. The boolean parameter true indicates that only connections with data available are returned. If the string array returned by this method is empty, the application was not started due to an incoming connection, although it might have been lauched by the AMS due to some other reason, such as a timer.

3.2 Web Services and SOAP

Web Services are services that utilise the web technology and use XML based communication over http. SOAP (Simple Object Access Protocol) is probably the most common XML-based technology for message exchange over HTTP. One of the fundamental ideas behind the web service paradigm is the use of standardized communication interfaces, usually based on the WSDL (Web Service Definition Language). This allows different clients to access these services in a uniform manner. The WSDL description can be used by the clients to build their communication with the server offering web services.

The J2ME Web Services API (WSA) is based on the JSR 172 and provides web service functionality for J2ME applications [7]. For communication with web services stub classes are needed in a MIDlet. Stub classes handle creation of outgoing SOAP messages and parse incoming messages. From the programmer's point of view using web services is calling suitable methods of the stub classes. With Sun Microsystem's Wireless Toolkit (WTK) stubs can be created using the WSDL document describing the web service.

3.3 Application deployment

Development time application testing is often done on a phone emulator as well as on a real device. The most effective way to deploy an application on a real device is probably through a dedicated software that communicates with the device through a USB cable, Infrared or Bluetooth. This allows the software developer to install an application to a device for testing and debugging. However, the end user does not always have the technical skills or equipment for deploying applications through a phone management software.

The Over The Air download (OTA) brings a solution for end user deployment of software for mobile devices [41]. In a typical scheme the user navigates to a download page using, for example, GPRS and clicks on a link to start the download. The device will first download the JAD (Java Application Descriptor) file, which tells the device important information about the application, such as, the size of the application and the intended Java configuration. The device can make decisions based on the details in the JAD file whether to continue download. If, for example, the device does not have enough space to accommodate the application, continuing the download is pointless.

After the application itself is downloaded, the device will install the application. This installation is device dependent and there is no general scheme for it. It should be noted that both the JAD file and the application JAR must have appropriate MIME (Multipurpose Internet Mail Extension) types attached to them for the download in order for the device to be able to handle them correctly.

Chapter 4

Secure SMS messaging

As it was stated earlier, SMS messages do not have adequate security for certain purposes that require confidentiality or integrity of data. Furthermore, authentication in GSM network is sometimes not adequate and there is practically no non-repudiation.

To create a solution that can provide the security measures for applications using SMS messaging we implemented java classes both for symmetric and asymmetric end-to-end encryption of SMS messages. These classes provide all four security measures; confidentiality, integrity, authentication and non-repudiation.

In the following sections we will cover implementation of end-to-end encryption for SMS messaging using both symmetric and public key encryption. Also, we cover how to implement strong two-way authentication, how to ensure integrity of data and authenticate the communicating parties.

There is, however, nothing in our applications that restrict them to communicate only via SMS. In fact a packet switched IP-based protocol over, for example, GPRS suits the applications better in some cases than SMS messaging. Enabling SMS, however, allows an application to be used by a wider audience. Also, an end-to-end security solution for voice in GSM was studied in [14].

4.1 Symmetric approach to SMS message encryption

Our first application (publication I) was targeted for the MIDP 1.0 platform. The requirements that were set on the application were the following:

1. Encrypt an SMS message using quasigroup encryption and send it

2. Receive encrypted messages and decrypt them

3. Operate fast enough to provide acceptable service in an environment with rather limited processing power

4. Be compact enough for use in mobile phone memory space

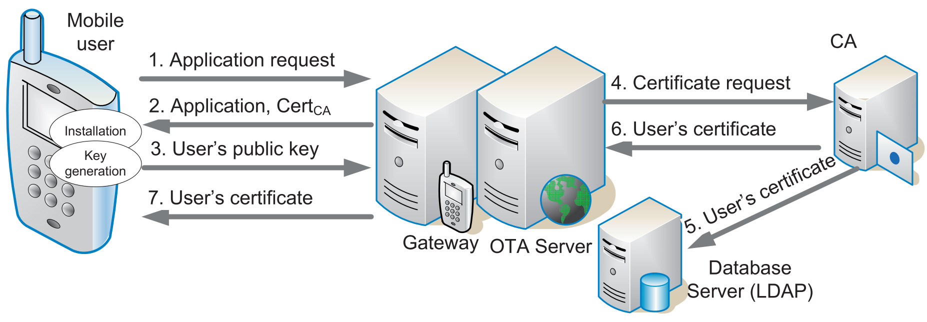

Figure 4.1: Application installation, key generation and certificate creation

Confidentiality and authentication were achieved through symmetric encryption using one of two encryption algorithms; Quasigroup or Blowfish. Blowfish is an iterated block cipher based on the Feistel structure [38] and was included into the study for comparison. Messages were sent and received in a binary format using the WMA.

4.2 PKI solution for SMS messaging

Our PKI solution presented in publication IV is a MIDlet, which can send and receive SMS messages in binary format using the WMA. We use RSA [46] algorithm for encryption and SHA-1 [46] for creating hash values. Although there have been reports (for example [51]) about problems in the security of SHA-1, we have adopted it since we consider it secure enough for our purposes. Changing the algorithm to another one, if need arises, is a straightforward task.

We use the Lightweight API from the Legion of the Bouncy Castle [24] which was designed for the J2ME environment. A J2ME application is a collection of byte code classes, which are packed into a JAR (Java Archive) file with some additional informative files. The usage of the Bouncy Castle generates very large JAR files, due to inclusion of classes from the library. We have used an obfuscator from proguard [23] to minimise the JAR file size. The obfuscator removes all unused class files from the JAR package making the package smaller. Obfuscation also makes reverse engineering of the class files more difficult by, for example, replacing class names with shorter ones. For developing the application we used Sun Microsystem's WTK (Wireless Toolkit) [45].

4.2.1 Installation

Figure 4.1 describes the download and installation procedure of our PKI solution. Before installing the software, details of the user have to be registered into the system. These details are later used to create a certificate for the public key of the user. We have used a Java Servlet with an HTTPS connection to enter the details of a user into a database (MySQL). During registration, the registration system generates a random nonce which is later needed in registration of the user's public key.

It is very important to reliably authenticate the user during registration. This is due to the fact that the user is granted a certificate later on based on the information provided during registration. We have adopted a strong authentication method based on a governmental PKI infrastructure. This infrastructure, the FINEID [34], can be used to produce legally binding digital signatures. The infrastructure is based on smartcards issued by the Population Register Centre in Finland [34]. To use the smartcard the client machine needs a client program such as the SETWEB [39] or the OpenSC [53].

However, there is nothing that ties the usage of the system to this authentication method. An organisation using our system can choose any authentication method they find appropriate.

Figure 4.1 shows an overall picture of the installation process. The software can be installed using the OTA (Over The Air) provisioning, see phase 1 in Figure 4.1. The device downloads the JAD (Java Application Descriptor) using an HTML/WML browser. The JAD gives some descriptive parameters of the application, such as required space on the device. Based of the information given in the JAD, the AMS (Application Management Software) will download a JAR file which contains all the classes needed to run the program. The AMS takes care of installing the application to the device. Along with the application, certificates of both the Certificate Authority CA and the Authentication Server AS are downloaded.

The first execution of the application will initiate a key generation procedure, where a 1024 bit private key and a corresponding public key are generated. The private key is stored on the device inside a PIN (Personal Identification Number) protected PKCS12 [36] container. The public key is also stored but without PIN protection. Storing of keys is done using the Record Store available in J2ME. The Record Store is guaranteed to maintain its content between power offs. An important observation in the key generation process is that the private key is never transmitted outside the phone. This ensures that a copy of this key cannot exist anywhere outside the phone.

The public key is sent to the server using an HTTPS connection, see phase 3 in Figure 4.1. The nonce created during user registration with the phone number of the user are needed to complete the certificate generation procedure.

4.2.2 Certificate creation

The server issues a certificate for the public key of the client. For generating the certificate we used the X509CertificateGenerator class of the Bouncy Castle. The certificate generation process is hence automated and requires no human interaction. The CA signs the certificate using the private key stored on the server. In order for the certificate to be verifiable in other devices, the CA certificate has to be issued by a known CA, whose certificate has been included in the device during manufacture. Often it is not possible by the user to add trusted root certificates into the device. The generated user certificate is stored in a file on the server and details of the user are saved in a database.

For the certificate generation, the client is authenticated with the phone number and a random nonce, which is generated during the registration of the user. We use a

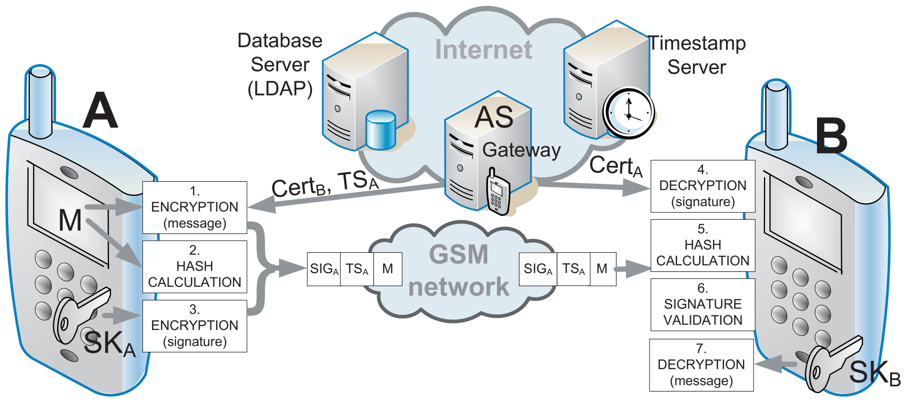

Figure 4.2: Secure message exchange between two mobile devices

protected HTTPS connection for registering both the user information and the public key. This insures that an adversary who can listen to the communication can not obtain the random nonce used in certificate creation. The client authenticates the server using the CA certificate that comes as a resource with the application package.

Usually certificates are stored in an LDAP (Lightweight Directory Access Protocol) directory. We are, however, storing the certificates on our server and provide an interface for retrieving generated certificates from our server. A phone number among other criteria can be used for retrieving certificates from our repository. This is important, since one may receive an SMS message with no other information about the sender except for the phone number.

4.2.3 Certificate retrieval

X509 certificates are usually stored in an LDAP directory, from which they can be queried. The FINEID certificates can be queried from ldap.fineid.fi. These certificates are needed when a mobile device wants to send a message to a recipient who uses a FINEID smart card. Retrieving a certificate from the LDAP directory with Java is quite straightforward, but we could not find an implementation of LDAP managing routines for J2ME. The options were either to port the routines to J2ME or to construct a mediator between our J2ME program and the LDAP directories. We chose the latter option and constructed a Java Servlet which gets user's details as parameters and searches an LDAP directory for a certificate (see VII for details). A further motivation for this approach was that a phone number of the user is not often included in a certificate, but we saw the phone number as a natural search criteria for finding a user's certificate.

For handling LDAP directories we used classes from the *com.novell.ldap* package [31]. These classes allow querying the directory and retrieving certificates.

4.2.4 Message Exchange

Fig. 4.2 describes the steps of user A sending a message to user B. A retrieves the certificate of B from the server if A does not have it from earlier communication. A must also verify the certificate and check that it has not been revoked. Since the CRL (Certificate Revocation List) can be quite large and the memory capacity of the mobile device is rather limited, we propose the usage of OCSP (Online Certificate Status Protocol) [29] for checking the status of the certificate.

In the rest of Section 4.2 we use the following notation: A and B are communicating mobile users and ID_X is a digital identifier of user X. In our system we use the FINUID identifiers, which uniquely identify users in the FINEID system. SK_X is an RSA private (secret) key of user X and PK_X is the corresponding public key. E_{SK_X} denotes private key encryption using the private key of user X and H denotes a hash function, SHA-1 in our case. TS_X is a timestamp generated by the timestamp server for user X. SIG_X is a digital signature created by X using the private key SK_X. $Cert_X$ is a certificate of user X and $OSCP_X$ is an OCSP token for a certificate of user X. We denote a message with M and a signed, timestamped message with MSG, $MSG = M|TS_X|E_{SK_X}(H(M|TS_X))$, where | denotes concatenation. The authentication server AS is used to get timestamps, OCSP tokens and certificates of other users.

Sending a message happens as follows:

1. $A \rightarrow AS : ID_B$
2. $AS \rightarrow A : TS_A, OCSP_B, Cert_B$ (optional), where both TS_A and $OCSP_B$ are signed by AS.
3. $A \rightarrow B : E_{PK_B}(M)|TS_A|E_{SK_A}(H(E_{PK_B}(M)|TS_A))$

Confidentiality of the message exchange is gained through A encrypting the message M with the public key PK_B of B. To ensure integrity and non-repudiation of the message, A digitally signs the encrypted message $E_{PK_B}(M)$ and the timestamp by calculating a hash value $h = H(E_{PK_B}(M)|TS_A)$ and signing it with the private key SK_A. Since A is the only one in possession of this key, A is also the only one capable of producing the signature $SIG = E_{SK_A}(h)$. If highly accurate time information is required, A can obtain a timestamp TS_A from the server. The timestamp is digitally signed by the server. When a timestamp is used, it is included in the hash value calculation of A's digital signature. Hence, A sends B $MSG = E_{PK_B}(M)|TS_A|E_{SK_A}(H(E_{PK_B}(M)|TS_A))$. Instead of the timestamp we can also use a sequence number. This means that the device does not have to contact the server, which means less network traffic.

4.2.5 Message reception

B receives a message $MSG = M'|TS'_A|SIG'_A$ from A, where M' is assumed to be the original message encrypted with B's public key, TS'_A is assumed to be the original timestamp TS_A and SIG'_A is assumed to be A's signature as described above.

B verifies A's signature SIG'_A by decrypting the signature with PK_A and comparing that to the hash value $H(M'|TS'_A)$. If B doesn't have A's certificate, B retrieves the certificate from the server. Again, using the OCSP, B can query the status of A's certificate. If the certificate proves to be valid, B can continue processing the message. B can also get a timestamp TS_B from the server and verify that TS'_A and TS_B are within an acceptable time window. If both the timestamp and the signature are valid, B concludes that the message came from A, was not tampered with and is recent, hence it is not a replay of some earlier communication. Only after a successful signature verification B proceeds to process the actual message.

B can recover the original message M by decrypting the received message M' with the private key, hence $E_{SK_B}(M') = M$, where SK_B is the private key of B.

In a nutshell, the whole procedure happens as follows:

1. $A \rightarrow AS : ID_B$
2. $AS \rightarrow A : TS_A, OCSP_B, Cert_B$ (optional), where both TS_A and $OCSP_B$ are signed by AS.
3. $A \rightarrow B : E_{PK_B}(M)|TS_A|E_{SK_A}(H(E_{PK_B}(M)|TS_A))$
4. $B \rightarrow AS : ID_A$
5. $AS \rightarrow B : TS_B, OCSP_A, Cert_A$ (optional)
6. B:
   (a) Check that $H(M'|TS'_A) = E_{PK_A}(SIG_A)$
   (b) Check that $TS_B < TS'_A + k$, where k is the predefined time window
   (c) $M = E_{SK_B}(M')$

The timestamps are used to avoid any replay attacks. The time window k for acceptable delay has to be short enough to be effective against such attacks, yet long enough to allow reasonable network delay. An additional security measure can be taken by the recipient B storing the last received timestamp of each sender A. The stored timestamp can be used to verify freshness of a timestamp the next time a message is received. This same measure is used with sequence numbers.

One SMS message can accommodate 140 bytes of data. With text messages this allows sending a total ot 160 characters when each character uses 7 bits. This means that in many cases more than one SMS message is needed to carry one protocol message. For digital signatures we use RSA algorithm, which means that the signature length is a multiple of RSA block size. Using RSA keysize of 1042 bits, one signature takes 128 bytes, only 12 bytes short of one SMS message.

4.2.6 User authentication

To authenticate a mobile user we have created a two-way authentication protocol based on the challenge-response scheme. An information system can send a challenge nonce N to the device using an SMS gateway. The message is assigned a port number, which states that the message is intended for our application. The AMS (Application Management Software) can automatically start the MIDlet upon receiving the message. Using a PIN code, the user can sign the challenge and send it back to the server. Obviously, it is not a very good idea to use ones private key to sign a random value on which one has no control. Rather, the client makes a calculation using the challenge and a random value the client has generated and signs the result. This result with the client's random value are sent to the server. By verifying the signature the server can be sure that the client is authentic. Also, the client can authenticate the server by verifying the signature the server sends.

Steps of the protocol for an information system IS sending a random challenge nonce N to a client C, C committing to this challenge and IS processing the commitment are as follows.

1. C initiates the protocol by asking for an access to a protected resource and offers ID_C and a random nonce N_C to IS.
2. $IS \rightarrow AS : ID_C$
3. $AS \rightarrow IS : OCSP_C, Cert_C$ ($OCSP_C$ is signed by AS and $Cert_C$ is optional)
4. $IS \rightarrow C : E_{PK_C}(N_{IS})|E_{SK_{IS}}(H(E_{PK_C}(N_{IS})|N_C))$, where N_{IS} is a random nonce generated by IS.
5. $C \rightarrow AS : ID_{IS}$
6. $AS \rightarrow C : OCSP_{IS}, Cert_{IS}$ ($OCSP_{IS}$ is signed by AS and $Cert_{IS}$ is optional)
7. $C \rightarrow IS : MSG = M|SIG_C$,where
   $M = E_{PK_{IS}}(N_{IS}|N_C)$ and
   $SIG_C = E_{SK_C}(H(E_{PK_{IS}}(N_{IS}|N_C)))$
8. IS:
   (a) $E_{SK_{IS}}(M) = N'_{IS}|N'_C$
   (b) If $H(N'_{IS}|N'_C) = E_{PK_C}(SIG_C)$
   $\Rightarrow N_{IS} = N'_{IS}$ and $N_C = N'_C$

The digital signature can be also used to show commitment to a contract between C and IS. Since the client is the only party in possession of the private key, this commitment is undeniable. The procedure would be similar to the one presented above, except that the nonce N_{IS} would be replaced by a contract CON_{IS-C} to which C would commit by signing $E_{SK_C}(H(CON_{IS-C}|N_C|TS_C))$. The procedure would in this case be initiated by IS, so the first phase could be omitted. However, some method is needed for C to authenticate IS.

Our system relies on SMS messaging in communications. Other means of communication are perhaps better suited for communicating with the server when one wants to retrieve certificates or OCSP responses. The need to consult the server can be omitted with designing the protocol for authentication by using the designated Authentication

Server AS. The certificates of both the CA and AS are downloaded with the application. The IS can request a signed and timestamped OCSP token $OCSP_{IS}$ from the CA or AS. This way the client C can verify the certificate of IS without opening a connection to CA or AS using the token and the certificate of CA.

4.3 PKI approach using PKI-SIM

Using a SIM card with PKI capability changes the scheme in a couple of ways. There is no longer a need to create keys on the device and the certificate has been created to the user by the system. Registration of user details can be done the same way, but obviously this has to be done before the card is issued to the user.

The PKI-SIM approach provides better security as the card is used as a secure cryptographic co-processor. This means that critical cryptographic processing is done on the card using the card's processor [10]. The secret key of the user is stored inside this tamper resistant card and it is practically impossible to extract the secret key from the card. Also, making copies of the card is infeasible.

In order to access extended features of the SIM-card (the security application), the J2ME environment requires an optional package, namely, *The Security and Trust Services API* (SATSA) [17]. Among other features, the SATSA specification defines methods for communication with applications on the SIM card, by exchanging messages in the APDU format [16]. There are some solutions that use the SIM card for strong authentication, but these solutions have been based on the SIM application toolkit (SAT) [11] and require SMS messaging between the card and an information system. These solutions do not facilitate third party applications to be developed.

We developed and tested our application in a partly simulated environment. We used a combination of:

- Real SIM cards with FINEID functionality;
- PC smart card readers;
- Simulated mobile phone with SATSA capabilities: SATSA Reference Implementation 1.0 [44] developed by Sun Microsystems, Inc.

FINEID public directory provides certificate search services using Lightweight Directory Access Protocol (LDAP) [31]. Since we did not find any implementations of LDAP for the Java 2 Micro Edition platform, we had to use one more mediator. A Java servlet accessible through an HTTPS server accepts certificate search requests, executes them using FINEID-directory LDAP server, and forwards search results (certificates) back to the caller. The servlet is also used for retrieving timestamps needed in protocols defined later.

4.4 EMS dispatch and Notification systems

Public healthcare has continuous pressure to cut costs as the population especially in developed countries is aging. One direction to look for savings is making health care more cost effective through use of information technology.

We found that two different ways have been used to utilise SMS messages in the health care domain [47]. Firstly, text messages were used to various reminders. Main point was to improve patients' compliance to the care at the appointed time, for example, arriving to the doctor or to get a vaccination [5, 50]. The other way to utilise SMS was to send information from patients home to the clinical information systems about, for example, different kinds of measurements done by a patient with diabetes [22, **?**]. How the security issues were dealt with was not mentioned in the articles.

European Union directive on privacy and electronic communications (2002/58/EC) gives regulations on protection of the privacy of individuals. The directive aims to harmonisation of regulations in Member States considering protection of fundamental rights and freedom. The directive sets limits on the collection and use of personal data in electronic communication [9]. Same kind of regulations are included in HIPAA (Health Insurance Portability and Accountability Act of 1996) regulations in USA [21]. Each country in Europe has its own, more exact, legislation regarding the handling of personal data in the health care domain. Main issue of those regulations is to protect individual's privacy, which is a fundamental base for confidential relationship between health care professionals and patients. We suggest a way to fulfil those requirements in case of using SMS messages in health care.

The ability to encrypt messages opens a wide variety of possibilities in several fields of health care. Our applications provide a safe storage for sent and received messages. When the messages include some delicate information such as patient data, it is very important that the messages in the phone cannot be read by outsiders. Currently there are solutions, where text messages are used, but there is no way to avoid a disclosure if the device is lost and the user has not deleted the received messages. Such systems include for example a system for sending assignment messages to emergency response teams, which in Finland often contain patients name and description of the emergency. Clearly this kind of messages should not be disclosed to a third party. There are various situations where patients or health care professionals would like to get a list of the patients' medication, for example, patients' appointment to the private sector or health care personnel in home care or in emergency situations. In case of asking information from health care providers' databases by SMS, it is essential to authenticate questioner by adequate means.

Our approach using symmetric encryption combines the phone number and the secret password to authenticate the user. Even in the case where an adversary can access the phone, he still does not have the necessary password. He can use any password, but in that case it is trivial to detect the fraud in the receiving end. In our case, where the EMS staff uses SSN (social security number) to retrieve the list of medication of a patient, the probability that a SSN encrypted with a wrong password decrypts to a valid SSN is extremely small, 10^{-19} (see publication III for details). The sending party (for example a hospital) will use a password registered between itself and the sender. Upon receiving the list of medications, the adversary is met with a task of decrypting the response. This is as hard as trying all the possible keys of the encryption algorithm, provided that the password is properly selected.

Using public key encryption, the authentication can be done using a digital signature. This approach would be compatible with the recent development of information security in health care.

Chapter 5

Mobile payments and protocols

In publication IX, following [20], we define mobile payments as wireless transactions of monetary value from one party to another where a mobile device (e.g., mobile phone, PDA, smartphone or any mobile payment terminal) is used in order to initialise, activate and/or confirm the payment. There are several ways to categorise mobile payments. Several different parameters, such as value, the transaction method or purchase type, can be used for categorisation. Publication IX summarises and categorises a number mobile payment systems.

The simplest mobile payment schemes on the market today rely on either calling or sending a text message to a premium number, which is later charged from the subscriber in the phone bill. The problem with this approach is the inadequate authentication and lack of non-repudiation. The user can claim that he did not make a particular call or send a certain message and the operator is faced with a very difficult task of proving the opposite. Without adequate security, the risks involved in mobile payments are too great for any operator to manage, and hence, most of these payment systems handle only micro payments essentially less than $20 of value.

A network operator usually concentrates on providing network services and does not have the capability to provide the service expected of a credit institution. For this reason, most mobile payment solutions use a mediator of some kind and this also creates additional cost.

There are PKI implementations of mobile payments on the market that solve the problem of non-repudiation. With these solutions transactions of larger values can be handled securely since they can provide non-repudiation. However, the solutions we found were implemented using the SIM toolkit and required operator involvement. Also, current (2006) implementations of the mobile FINEID are tied to the operator due to a proprietary encryption mechanism in the communication between the card and the authorisation server. This ties the solution to a specific network operator and to a specific credit organisation (issuer). Our solution is based on a PKI system administered by the government. In essence the main difference is that our system authenticates citizens instead of customers of a specific credit institution or network operator. This makes the system open to users of any operator or credit institution.

This chapter summarises the protocols we have written for mobile payments. The secure message exchange described in the previous chapter is a building block that is necessary for the payment protocols to be realised. We describe two different protocols; One for virtual point-of-sale (POS) and another one for real point-of-sale. By real POS we mean a situation, where the buyer is physically present whereas virtual POS means a situation where the seller and the buyer do not physically meet.

5.1 Payment protocols

Our mobile payment scheme includes the following parties. A *customer* is a user of a mobile hand-held device. The customer has received a SIM card with the FINEID applet, which includes the public key certificate of the user and a corresponding secret key. Identity of the customer is their Finnish Electronic User ID (FINUID).

A *merchant* is an owner of a point-of-sale terminal (or a vending machine) or a service provider that accepts mobile payments. The merchant has a secret key and a corresponding public key certificate registered in the FINEID system.

A *bank* or another credit organisation like VISA or MasterCard is a financial institution that acts as a payment processor. The customer has an account in the bank, or has been issued with a credit card operated by it. If the customer has multiple accounts or credit cards within the bank, the bank has been informed which of them should be used for mobile payments. The bank has the right to charge the customer's account or credit card when presented with a payment order signed with the customer's secret key.

In this section we describe two mobile payment protocols: one for a virtual POS payment, and another for a real POS (vending machine) payment. The following notation is used in the description: C is a customer, M is a merchant, and B is a bank. ID_X is the identity of subject X. SK_X is the secret RSA key of subject X, and PK_X is the corresponding public key. $Cert_X$ is the public key certificate of subject X. $\{m\}_K$ denotes RSA encryption of the message m under the key K. SIG_{X_Y} is a digital signature generated by X, intended to be verified by Y. H is a hash function; we use SHA-1 in our protocol.

5.2 Virtual POS Payment

The protocol for a mobile virtual POS payment contains the following steps (see Fig. 5.1):

Service request. In phase 1, the customer initiates the protocol with the merchant by requesting product options. The request may contain information which limits possible options, such as date and time.

$$C \xrightarrow{\text{Service request}} M$$

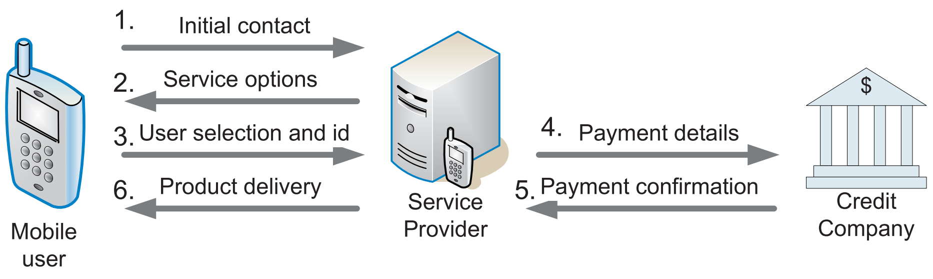

Figure 5.1: Virtual POS Payment Model

Service options. In phase 2, the merchant sends a list of options to the mobile device. The list includes short descriptions of products and pricing information. The merchant also attaches its certificate to the list of options.

$$M \xrightarrow{\text{Service options}|Cert_M} C$$

Product selection. In phase 3, the customer is prompted by the mobile device to select a product from the list. The information on the customer selection is sent to the merchant. The selection is signed using the private key of the customer. The message is

$$C \xrightarrow[\mathrm{SIG}_{\mathrm{C_B}}=\{H(TS_C|ID_M|AM)|H(PD|N_C)\}_{SK_C}]{\mathrm{MSG}=\left\{PD|N_C|TS_C|\{H(PD|TS_C)\}_{SK_C}|\mathrm{SIG}_{\mathrm{C_B}}\right\}_{PK_M}} M$$

where PD is a string that describes product details, N_C and TS_C are a random nonce and a timestamp generated by the customer C and AM is the amount of money the purchase will cost.

Payment request. Phase 4 of the protocol includes the merchant sending the payment details to the credit company. This payment information is signed using the merchant's private key and encrypted using the the public key of the credit company. The message in phase 4 includes merchant's details and payment details, such as amount, id of the customer and the signed message received in phase 3.

$$M \xrightarrow[\mathrm{SIG}_{\mathrm{M_B}}=\{H(ID_M|ID_C|TS_C|AM|H(PD|N_C))\}_{SK_M}]{\mathrm{MSG}=\left\{ID_M|ID_C|TS_C|AM|H(PD|N_C)|\mathrm{SIG}_{\mathrm{M_B}}|\mathrm{SIG}_{\mathrm{C_B}}\right\}_{PK_B}} B$$

In this message SIG_{C_B} is the same signature as in phase 3. After receiving this message the credit company B checks that the timestamp TS_C is newer than the timestamp of the previous communication to detect any replay attacks.

Payment confirmation. The indicated amount of money is transferred from the account of the buyer to the account of the seller. Phase 5 is initiated by the credit company if this transaction can be processed and finalised. The credit company sends a confirmation message to the merchant. The message is signed using the private key of the credit company.

$$B \xrightarrow{\text{MSG}=\{H(ID_M|ID_C|TS_C|AM|H(PD|N_C))\}_{SK_B}} M$$

From this message the merchant can check that the payment was made with the agreed amount AM and from the account of C to the account of M. The hash value $H(PD|N_C)$ is meant for the customer to make sure that the merchant can not claim that the customer bought something else than the original product.

Product delivery. Finally, in phase 6, the merchant checks the message received in phase 5. If the message is valid and the payment has been done, the merchant delivers the product to the customer. The merchant also sends the customer a message stating that the payment has been made and the product has been delivered.

$$M \xrightarrow{\text{MSG}=\{H(ID_M|ID_C|TS_C|AM|H(PD|N_C))\}_{SK_B}} C$$

The customer can check that the amount of money AM, product details PD, the nonce N_C and the timestamp all match the original values to be sure that he paid the correct amount for the correct product.

Use case: Buying a train ticket

Our protocol could be used, for example, to buy tickets for public transportation. These tickets are sometimes of considerable monetary value. At the same time, it is expensive to maintain ticket sales offices. Buying a train ticket would happen as follows:

1. The customer inquires for journey options by sending an SMS message to the train operator. The message includes the starting point, the destination point, date, and, perhaps, approximate time of the planned journey.
2. The train operator responds with all possible connections matching the query, along with pricing information.
3. The customer selects a suitable journey and sends a request to the train company. The request includes an identifier of the chosen connection and customer's credit account details. The credit account details are encrypted with the public key of the credit company, and hence are not readable by the train company. The order is signed by the customer.
4. The train company sends the credit company the payment amount and its details signed and encrypted with the public key of the credit company. Also, the credit account and payment order details sent by the customer are included in this message.

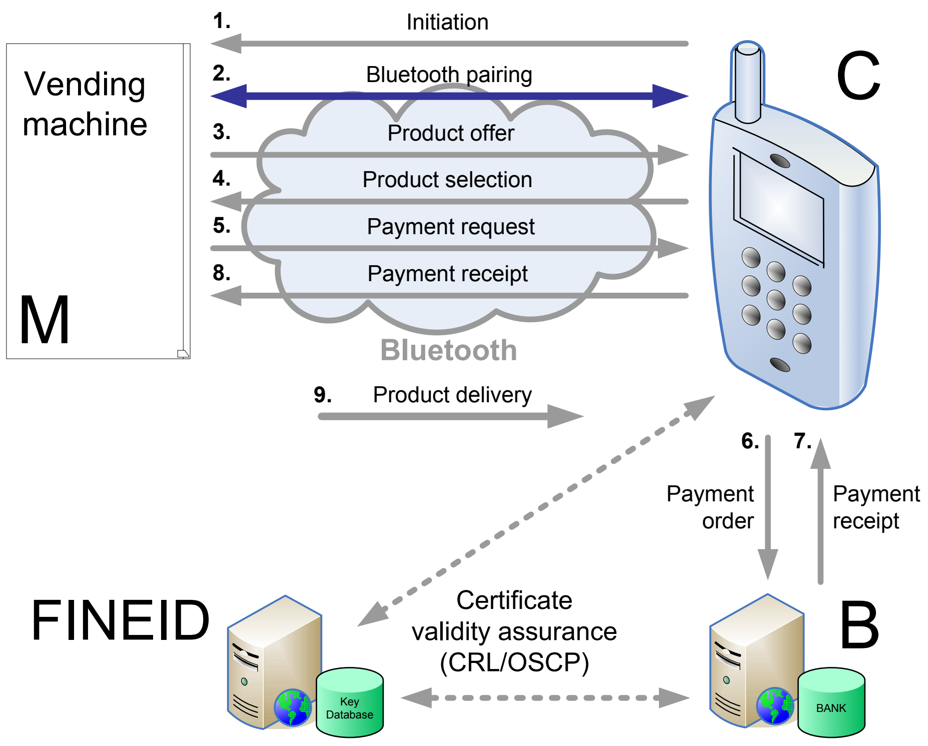

Figure 5.2: Protocol overview

5. After receiving the message from phase 4, the credit company checks the signatures of both the customer and the train company, debits the customer's account and sends a reply to the train operator. The reply contains a payment receipt signed by the credit company.

6. The train company issues a digitally signed electronic ticket. The ticket is sent to the customer along with the payment receipt. In the train, the conductor can verify the electronic ticket, if necessary, by a code stored in the train company's information system.

5.3 Real POS (Vending Machine) Payment

Our protocol for a secure vending machine payment contains the following steps (see Fig. 5.2):

Initiation. In phase 1, the user C initiates the protocol with the merchant M by choosing a product. In case the vending machine supports several ways of payment, the user may need to explicitly select the mobile payment option. Optionally, the protocol can be initiated by the vending machine, which detects the device when it comes in the range of the Bluetooth communication. No messages are sent in this phase.

Bluetooth pairing. To enable exchange of messages, Bluetooth pairing must be performed between the vending machine and the mobile phone. In case there is a possibility that several devices are in the range, the machine can use a random PIN code for pairing and show this PIN on its display. User must enter this PIN code in the mobile phone.

Bluetooth pairing. If the user has not selected a product yet, the vending machine sends a message with information about available products and their prices. In case phase 1 was initiated by the user, and the product is already selected, the list of products contains only the selected item. In addition to this data, the vending machine sends its own certificate $Cert_M$ and a random nonce N_M.

$$M \xrightarrow{\text{MSG}=Cert_M|N_M|\texttt{List of products}} C$$

After receiving the message, C extracts M's certificate and checks its validity.

Product selection. The user is prompted by the mobile device for selection of a product, unless it has already been selected. The information on the user selection is sent to the vending machine. Also, the certificate $Cert_C$ holding the user's public key is sent to the device.

The mobile phone must store the price AM of the selected product, as it will be needed later for payment.

The message in phase 4 consists of three parts. The first part is the user selection S, and a nonce N_C generated by the mobile device. This part of the message is encrypted with the vending machine's public key PK_M. Second, the user's certificate $Cert_C$ is appended to this message. The last part of the message is a signature $SIG = \{H(S|N_M|N_C)\}_{SK_C}$.

$$C \xrightarrow[\text{SIG}=\{H(S|N_M|N_C)\}_{SK_C}]{\text{MSG}=\{S|N_C\}_{PK_M}|Cert_C|\text{SIG}} M$$

After receiving the message, M extracts C's certificate, verifies it, and decrypts the signature SIG obtaining $h = H(S|N_M|N_C)$. After this M decrypts the message MSG obtaining S and N_C. To conclude, M calculates the hash $h' = H(S|N_M|N_C)$ and verifies that $h' = h$. The vending machine could also check the certificate revocation list to see that the user certificate has not been revoked, but this checking can also be made responsibility of the bank.

Payment request. In phase 5 of the protocol the vending machine sends a payment request to the mobile device. The request is signed using the vending machine's private key SK_M.

The payment details include the account number ACN_M, and a reference id of the vending machine ID_M. Note that the price of the product is not sent with the payment details, since C already knows it. However, it is included in the second part of the message. Namely, C's certificate, price of the product AM, and two nonces N_M and N_C are concatenated, hashed, signed with the vending machine's private key and appended to the message MSG. The last part of the message is a signature $SIG = \left\{H(ACN_M|ID_M|\{H(Cert_C|AM|N_M|N_C)\}_{SK_M})\right\}_{SK_M}$.

$$M \xrightarrow[\text{SIG=}\left\{H(\mathrm{ACN_M|ID_M|\{H(Cert_C|AM|N_M|N_C)\}_{SK_M}})\right\}_{SK_M}]{\text{MSG=}ACN_M|ID_M|\{H(Cert_C|AM|N_M|N_C)\}_{SK_M}|\text{SIG}} C$$

C will later send the signed hash of C's certificate, price and both nonces to the bank B. The bank will use (and optionally store) it as a proof of transaction authorisation by vending machine. This way C can not offer one certificate to M and another to B, or change the amount to be paid. The signed hash can also be stored by C as a receipt from the vending machine. Combined with a receipt from the bank (see phase 7), it can be used later as a proof of purchase if a dispute arises.

After receiving the message MSG, C verifies the signature SIG using M's public key.

Creation of a payment order. In phase 6 the mobile user C communicates with the bank B. In addition to the information received from M in the previous steps, an account number of the mobile user ACN_C and a timestamp TS are needed for the transaction.

C creates a payment order
$PO = ACN_C|ACN_M|ID_C|ID_M|AM|N_M|N_C|TS|H(Cert_C|AM|N_M|N_C))_{SK_M}$.
The payment order is sent to the bank encrypted with the bank's public key and signed with the C's private key.

$$C \xrightarrow[\text{MSG=}\{PO\}_{PK_B}|\text{SIG; SIG=}\{H(PO)\}_{SK_C}]{PO=ACN_C|ACN_M|ID_C|ID_M|ID_B|AM|TS|\{H(Cert_C|AM|N_M|N_C)\}_{SK_M}} B$$

In this message everything except C's account number ACN_C was received from M in the previous stage. The bank B is obviously the one in which the mobile phone user has an account.

Here we assume that the mobile device already has the public key of the bank. Certificates of participating banks can be installed into the device when the software is installed. We can also include a procedure for importing a certificate of a bank which has joined the protocol after the software was installed.

Payment processing. After receiving and decrypting the payment order, the bank verifies C's signature attached to it. For this, the bank retrieves C's certificate from the FINEID LDAP (Lightweight Directory Access Protocol) directory; ID_C is used as the search key. In the same way the bank gets M's certificate in order to verify the signature $\{H(Cert_C|N_M|N_C))\}_{SK_M}$. This is done to make sure that the same certificate and nonces were used in communication between C and M. In addition, the bank checks that neither of the certificates have been revoked. The bank also compares the timestamp TS to the stored timestamp of the previous payment order received from C (if any) to defeat replay attacks. Upon successful pass of all checks, the bank transfers the amount of money from C's account to M's account. In case M's account is in another bank, usual interbank procedures are used for crediting money to M. If the transaction can be processed and finalised, the bank initiates the phase 7, where it encrypts and sends a confirmation message (receipt) to the mobile phone.

The receipt provides a proof that the payment has been made. The bank account number of the vending machine, amount of money and nonces N_M and N_C are hashed and signed using the bank's private key:

$$B \xrightarrow{\text{MSG}=\{H(ACN_M|AM|N_M|N_C)\}_{SK_B}} C$$

C has all the information needed for calculation of the same hash and verification of the bank's signature.

Proof of payment. In phase 8, the mobile phone forwards the bank receipt to the vending machine. In order to specify which bank's public key must be used for verification of the receipt, the bank's id ID_B is included in the message.

$$C \xrightarrow{\text{MSG}=ID_B|\{H(ACN_M|AM|N_M|N_C)\}_{SK_B}} M$$

We assume that the vending machine already has certificates of participating banks. Therefore, the vending machine can decrypt the receipt using the bank's public key. The vending machine then calculates hash $H(ACN_M|AM|N_M|N_C)$ and verifies that its value is the same as in the receipt.

The vending machine must have a list of valid public key certificates of different banks. The vending machine does not need to have a network connection, the protocol may be extended to check the validity of bank certificates by forwarding Online Certificate Status Protocol (OCSP) requests through the mobile phone to a trusted OCSP server.

5.4 Traffic analysis

Analyzing the amount of traffic in the protocols is not quite straightforward as there are numerous factors affecting it. When using public-key encryption, the keysize directly affects the amount of data needed for one protocol message. With a 1024 bit (128 byte) keysize one block of ciphertext is 128 bytes in length, meaning that one digital signature will need 128 bytes.

In section 4.2.4 A sent a message to B as follows:

1. $A \rightarrow AS : ID_B$
2. $AS \rightarrow A : TS_A, OCSP_B, Cert_B$ (optional), where both TS_A and $OCSP_B$ are signed by AS.

3. $A \to B : E_{PK_B}(M)|TS_A|E_{SK_A}(H(E_{PK_B}(M)|TS_A))$

Obviously, the first message of the protocol can be delivered as one binary message. The second message of the protocol may require more than one SMS message, depending on keysize. The timestamp TS_A takes one RSA block since it is signed; the same is true also for the OCSP response $OCSP_B$. The time information can be of variable length, depending on the accuracy requirements. A considerably accurate timestamp of 12 bytes fits well in a 140-byte long SMS message, along with a signature (assuming that a 1024-bit long RSA key is used). Similarly, the OCSP response (cerificate serial number and status byte) fits into 12 bytes, so the timestamp and the OCSP response together take two SMS messages. The optional certificate $Cert_B$ can have a varying size, but commonly it will be near a thousand bytes. Although the certificate needs to be acquired only once, rather than transmitting it through SMS we propose using a packet data connection or preinstalling certain certificates with the application.

In the third message, the length of the first part $E_{PK_B}(M)$ obviously depends on message M and is a multiple of the cipher block size. In general, the length of $E_{PK_B}(M)$ is $\lceil \frac{|M|}{b-1} \rceil b$, where b is the RSA cipher key size in bytes and $|M|$ is the message length in bytes. The length of the third part equals the cipher block size, as the hash value is shorter than the cipher block size. The protocol message size requirements in other protocols can be analyzed similarly.

Chapter 6

Conclusions and future work

SMS is probably the most used data service in the GSM network and it is used in vast amount of applications, such as reminder or dispatch systems. In cases where confidentiality is required, symmetric end-to-end encryption can be used. As this service is not built in devices, an application can be installed on the device to achieve this functionality.

With the increase of capable devices, the number of mobile services have increased steadily. Modern devices can offer a wide range of services including mobile commerce and mobile government services. These services obviously require security services as users expect confidentiality and correctness of transactions. At the same time service providers expect reliable authentication of users and non-repudiation of transactions.

Today, the most common method of payment in electronic commerce is a credit card. A fraud in payment using an magnetic stripe credit card is a real risk, since with magnetic stripe credit cards authentication of the credit card user is poor. Non-repudiation in current systems is practically impossible to achieve, since the credit card user surrenders all of the authentication credentials to the seller. A PKI system can solve the problem as the digital signature can be created only by the owner of the private key. Credit cards, such as the EMV (Europay-Mastercard-Visa) system that use a chip instead of the magnetic stripe, have overcome these problems.

Among users, SMS has superior availability in comparison to networked computers or credit cards. People tend to carry their mobile phone with them so that it is available more often than a computer with a network connection. This makes the market for mobile payment through mobile phone superior to the above mentioned alternatives.

6.1 Symmetric encryption

Our application for symmetric end-to-end encryption (publication I) provides confidentiality of messages. It can furthermore be used in authentication and access control, since we assume that the shared secret is known only by authorized parties. The phone book relieves the user from remembering different passwords shared with different users. Storing contacts in the phone book is safe, since they can be accessed only with the user password.

The novelty of this approach was in the ability to send and receive encrypted messages between mobile devices. Also, the application can be installed to a wide range of mobile devices that otherwise have no messaging security solution.

In applications, where access control is vital, our application can be used to authenticate the sender of a message. The phone number and the password can be used as credentials, like in any password based authentication. Also it is possible to detect, if the message has been corrupted or tampered with during transmission. Most importantly, the messages containing delicate information are stored securely and remain undisclosed even when the device is accessed by an adversary.

6.2 Public key encryption

Symmetric encryption can not provide a solution to the sometimes crucial requirement of non-repudiation. For this reason the application that uses symmetric key encryption is not suitable for delivering security to areas like mobile payments.

The public key encryption scheme we described provides strong authentication of communicating parties, non-repudiation, integrity and confidentiality. Non-repudiation is especially important in situations where a message is used to show commitment, for example, to an order. Using PKI we can be sure that a signature was really made by the person in possession of the private key. Since the keys are created within the cryptographic token and the private key is never exposed outside it, we can be sure that a valid signature of a user cannot be forged.

Strong user authentication is a crucial requirement for the development of commercial mobile services. The use of governmentally administered PKI can significantly increase trust in modern electronic commerce, to the benefit of both service providers and their customers.

We see a lot of potential in using our applications in a wide range of mobile services. SMS technology is a mature and well-tried service in the GSM network and has superior availability in GSM networks globally. Using PKI with OCSP generates more traffic and in the last two publications (VIII and IX) the communication was based on packet data (UMTS or GPRS, depending on availability). Combining these communication methods we can obtain a very good platform to develop services for mobile authentication, access control, e-commerce and other areas where secure communication is needed.

6.3 Payment protocols

Based on a public key infrastructure we proposed a novel mobile payment scheme. The scheme can be used in real POS (Point-Of-Sale) as well as in virtual POS. The main advantage of our system is that it does not require any mediator. This reduces the total cost of a payment. We also described two protocols, one for virtual POS and another one for real POS.

Our system utilizes a governmental PKI infrastructure, namely the FINEID, making it an affordable solution since the administration of the system is provided by the government. Furthermore, as citizens have adopted this system for secure electronic transactions, it has a high level of trustworthiness. Our system is built using Java to gain the best possible portability across device platforms.

Our proposed MP solution provides strong two-factor authentication of communicating parties, integrity of data, non-repudiation of transactions, and confidentiality of communication. By using a governmentally administered PKI system we authenticate citizens instead of customers of a certain operator or credit institution. This makes the system open to all merchants, financial institutions and mobile users.

6.4 Future work

One of the optimatizations we did not consider in our papers is compression of messages when using SMS in communication. Compressing messages before encryption might make messages shorter when symmetric encryption is used. With asymmetric encryption, the encrypted part has a fairly uniform length that depends on the modulus size. With encrypted messages compression is not really an issue as encrypted data should exhibit a rather random structure. If the encrypted data shows regularities, it may be used to attack the cipher.

As stated earlier, in our PKI implementations for mobile payments (publications VIII and IX) we used packet data as our communication solution. Usage of SMS messaging in this scenario seems also quite possible, but further study concerning the reliability and cost would be in order. Certain changes should be done since, for example, the author's FINEID certificate is more than 1400 bytes in size meaning more than ten messages if transferred via SMS. Using only SMS, a number of certificates should be installed with the application.

The Quasigroup encryption is a rather new encryption method that has not been studied very much by the cryptographic community. There is still a lot to do on the security analysis of this cipher. From the implementational aspect one interesting challenge would be to be able to generate a Latin square based on a password. This mapping would enable to use a password as a shared secret instead of using latin squares as secrets. Currently a password can be used as a shared secret dictating the selection of leaders, primitives and latin squares as shown in the Fig. 2.1.

One very interesting usage of Quasigroup encryption could be implementing whitebox cryptography [6]. It seems that as the nature of Quasigroup encryption is essentially based on table lookup, implementing whitebox cryptography would be more straightforward than with block ciphers, such as DES or AES.

NFC (Near Field Communication) is an RFID (Radio Frequency Identification) based technology for short-range wirelss communication that has been used for example in smart posters and Bluetooth fast pairing. The communication range is typically 0-20 centimetres facilitating touch-to-launch applications [12]. NFC could be used to expedite the Bluetooth pairing phase of the real POS payment scheme presented in 5.3. At the moment (23.10.2007) there are a couple of NFC enabled phone models on the market and the penetration of NFC enabled mobile phones is estimated to grow rapidly

[1]. It is also possible to have data interchange between two NFC devices so that both devices generate their own radio fields (also called the active operation mode) in comparison to the passive mode in which only the reader/writer device generates the radio field and the tag merely responds. This implies the usability of NFC communication as a transfer media in protocol 5.3 instead of Bluetooth.

In the message exchange in 4.2.4 timestamps are used to show the freshness of messages. This is done mainly to protect users against replay attacks. It can also later be used to demonstrate the time a message was created. This type of timestamp does not, however, protect a user if the user's private key is compromized. An enhancement, where the user sends a hash of the message to the authentication centre could be used. This way the timestamp would be tied to a particular message and even in the situation where the private key is compromized, it would be possible to determine whether the message was created before or after the key was blacklisted.

The current implementation of the FINEID on the PKI-SIM ties the user to the network operator as it uses a SAT applet that communicates with outside systems through encrypted SMS messages. This encryption is proprietary and hence the applet cannot be used by third parties. In order to open the system for third party application developers, the scheme should be modified so that the PKI-SIM would contain the same FINEID application as the ID cards. Cooperation between the PRC and network operators would be necessary to realize this.

Bibliography

[1] ABI Research: Twenty percent of mobile handsets will include near field communication by 2012. London, UK (April 2007, accessed 23.10.2007) http://www.abiresearch.com/abiprdisplay.jsp?pressid=838.

[2] Barkan E., Biham E., Keller N.: Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication
Proceedings of Crypto 2003
Available (21.4.2007) on http://www.cs.technion.ac.il/∼biham/publications.html

[3] Biham E.: Differential Cryptanalysis.
http://www.cs.technion.ac.il/∼biham/

[4] Biham E., Shamir A.: Differential Cryptanalysis of DES-like Cryptosystems
Journal of Cryptology, vol 4, no. 1, 1991

[5] Car J., Sheikh A.: Email consultations in health care: 1 - scope and effectiveness. BMJ. 2004: 329: pp. 435-438 (4).

[6] Chow S., Eisen P., Johnson H., van Oorschot PC.: A White-Box DES Implementation for DRM Applications
Proc. 2nd ACM Workshop on Digital Rights Management, 2002 - Springer LNCS

[7] Ellis J., Young M.: J2ME Web Services 1.0,
Sun Microsystems Inc. 2003
Available (25.4.2007) on http://jcp.org/en/jsr/detail?id=172

[8] European Commission: Summary of positions on an application of article 3.3.e to combat theft of mobile phones. Brussels, 12 February 2004

[9] European Union: Directive on privacy and electronic communications 2002/58/EC. (10 Jan 2005.) Available at http://europa.eu.int/eur-lex/pri/en/oj/dat/2002/l_201/l_20120020731en00370047.pdf.

[10] Gemplus: About Smart Cards, Related Technologies Gemplus 2005. Available (4.4.2005) on http://www.gemplus.com/smart/cards/tech/#PKI

[11] Gemplus: SA.: SIM Toolkit
http://www.gemplus.com/techno/stk/

[12] Innovision Research & Technology plc: Near Field Communication in the real world. Turning the NFC promise into profitable, every day applications. (accessed 23.10.2007) http://www.innovision-group.com/scripts/doc_view.php?docID=7

[13] Gomez E, Hernando M, Garcia A, Del Pozo F, Cermeno J, Corcoy R, Brugues E, De Leiva A.: Telemedicine as a tool for intensive management of diabetes: the DIABTel experience. Computer Methods and Programs in Biomedicine. 2002: 69: pp. 163-177(15).

[14] Hassinen M.: Fast End-to-End Encryption for Voice in GSM Networks.
In Communication Systems and Networks, Prodeedings of AsiaCSN 2007. Phuket, Thailand. Acta Press, 2007.

[15] Hassinen M.: SafeSMS 1.0 user manual. October 2004, Department of Computer Science, University of Kuopio. http://www.cs.uku.fi/∼mhassine/SafeSMS/Manual_en.pdf

[16] ISO/IEC 7816-4:1995.: Integrated circuit(s) cards with contacts. Part 4: Interindustry commands for interchange.

[17] Java Community Process: JSR-000177 Security and Trust Services API for J2ME. http://jcp.org/aboutJava/communityprocess/final/jsr177/

[18] Java Community Process: JSR-000118 Mobile Information Device Profile 2.0.
http://jcp.org/aboutJava/communityprocess/final/jsr118/

[19] Java Community Process: JSR-000120 Wireless Messaging API. http://jcp.org/aboutJava/communityprocess/final/jsr120/

[20] Karnouskos S., Mobile payment: A journey through existing procedures and standardization initiatives
IEEE Communication Surveys 6 (4) (2004) 44–66.

[21] Kibbe D. HIPAA privacy and security regulations. The Case Manager. 2001: 12: pp. 34-39(6).

[22] Kwon H-S., Choa J-H., Kima H-S., Lee J-H., Song B-R., Oh J-A., Han J-H., Kim H-S., Cha B-Y., Lee K-W., Son H-Y., Kang S-K., Lee W-C., Yoon K-H.: Development of web-based diabetic patient management system using short message service (SMS). Diabetes Research and Clinical Practice. 2004: 66 : pp. 133-137(5).

[23] Lafortune E.: Proguard web pages. http://proguard.sourceforge.net/

[24] Legion of the Bouncy Castle: Lightweight API
http://www.bouncycastle.org/

[25] Markovski, S.: Quasigroup string processing and applications in cryptography: Proceedings of the Conference MII 2003, 14 - 16 April, Thessaloniki, Greece (pp. 278-290)

[26] Markovski, S., Gligoroski, D., and Bakeva, V.: Quasigroup String Processing: Part 1, Contributions, Sec. Math. Tech. Sci., MANU, XX 1-2(1999) 13-28

[27] Markovski, S., Kusakatov, V.: Quasigroup String Processing: Part 2, Contributions, Sec. Math. Tech. Sci., MANU, XXI 1-2(2000) 15-32

[28] Mehrotra A., Golding L.: Mobility and Security Management in the GSM System and Some Proposed Future Improvements.
Proceedings of the IEEE, vol. 86. no 7, july 1998.

[29] Myers M., Ankney R., Malpani A., Galperin S. and Adams C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP
http://www.ietf.org/rfc/rfc2560.txt

[30] m-travel.com: Show tickets sold on mobile phone in Singapore. July 31, 2003
Available (21.4.2007) on http://www.m-travel.com/news/2003/07/show_tickets_so.html

[31] Novell corporation: LDAP documentation
http://developer.novell.com/ndk/doc/jldap/jldapenu/ietfapi/index.html

[32] Open Mobile Alliance: Wireless Identity Module. Available (21.4.2007) on http://www.openmobilealliance.org/release_program/docs/obkg/v1_0-20050322-c/oma-wap-wim-v1_2-20050322-c.pdf

[33] Pesonen L.: GSM Interception.
Helsinki University of Technology, 1999
Available (22.4.2007) on
http://www.tml.tkk.fi/Opinnot/Tik-110.501/1999/papers/gsminterception/netsec.html

[34] Population Register Centre: FINEID-S4-1 Electronic ID Application
Population Register Centre 2004. http://www.fineid.fi

[35] Population Register Centre: FINEID-S1 Implementation Profile for Finnish Electronic ID Card
Population Register Centre 2004. http://www.fineid.fi

[36] RSA Laboratories: PKCS #12: Personal Information Exchange Syntax Standard, version 1.0.
RSA Security Inc. 1999. Available (24.4.2007) on
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-12/pkcs-12v1.pdf

[37] RSA Laboratories: PKCS #15 v1.1: Cryptographic Token Information Syntax Standard.
1998-2000 RSA Security Inc. Available (21.4.2007) on
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.doc

[38] Schneier B.: Applied Cryprography. Protocols, algorithms and source code in C.
John Wiley and Sons, Inc. 1996.

[39] Setec: SETWEB
http://www.setec.fi/

[40] Siemens: SMTK Programmer’s Reference Manual, Version 4.2.
Siemens AG, 10/2004. http://communication-market.siemens.de/

[41] Siemens: OTA Specification Release 1.0.2
http://communication-market.siemens.de/

[42] Sun Microsystems, Inc.: Java ME Technology APIs & Docs.
Available (21.4.2007) on http://java.sun.com/javame/reference/apis.jsp

[43] Sun Microsystems Inc.: J2ME API Documentation.
http://java.sun.com/

[44] Sun Microsystems, Inc.: Reference Implementation of the SATSA specification ver. 1.0.
http://java.sun.com/products/b/

[45] Sun Microsystems: User Guide, Wireless Toolkit, Version 2.0. Java Platform, Micro Edition.
http://java.sun.com/

[46] Stallings W.: Cryptography and network security, Principles and Practice.
Prentice-Hall, 1999.

[47] Treweek S.: Joining the mobile revolution
Scandinavian Journal of Primary Health Care, 2003;21:75-76

[48] Vaudenay S.: On the weak keys in Blowfish.
Proceedings of the 3rd International Workshop of Fast Software Encryption.
Springer Verlag, 1996, pp. 27-33.

[49] Vedder K.: GSM:Security, Services and the SIM.
State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography, Leuven, Belgium, June 1997.
Volume 1528/1998, Springer-Verlag

[50] Vilella A., Bayas J-M., Diaz M-T., Guinovart C., Diez C., Simo D., Munoz A., Cerezo J.: The role of mobile phones in improving vaccination rates in travelers. Preventive Medicine. 2004: 38: pp. 503-509(7).

[51] Wang X., Yiqun Y, Yu H.: Collision Search Attacks on SHA1.
Published Feb 13 2005. Available (24.4.2007) on http://theory.csail.mit.edu/ yiqun/shanote.pdf

[52] Wireless Application Protocol Forum, Ltd.: Wireless Identity Module Part: Security 12-July-2001. Available (21.4.2007) on www.wapforum.org/tech/documents/WAP-260-WIM-20010712-a.pdf

[53] Yrjölä J., Teräs T., Tapaninen A.: OpenSC project http://www.opensc-project.org/