A study of user authentication using mobile phone

Steffen Gullikstad Hallsteinsen

Master of Science in Communication Technology

Submission date:	June 2007
Supervisor:	Van Thanh Do, ITEM
Co-supervisor:	Ivar Jørstad, Ubisafe AS

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Originally the mobile phone is intended to be a device that enables communications between people. However, due to its connectivity and security functions combined with its ubiquity, the mobile phone's usage can be extended to be a security token used in the authentication of the user for all types of services. A previous project [1] has identified several solutions which utilise the mobile phone and the SIM for authentication towards services. These solutions use different communication channels and patterns in order to securely establish the identity of the user (e.g. through SMS, through Bluetooth and Internet etc.). The goal of this thesis is to analyse, design, implement and evaluate one (or if time permits, several) of these authentication solutions using the mobile phone and its SIM card as authentication tokens.

The thesis work consists of the following tasks:

a. Study of required background information
b. Brief study of possible authentication solutions
c. Selection of which solution(s) to implement
d. Analysis, design and implementation of selected solution(s)
e. A thorough evaluation of the security properties of the implemented solution(s)

Assignment given: 22. January 2007
Supervisor: Van Thanh Do, ITEM

Preface

This thesis is done as the final work in the Master of Technology program at the Norwegian University of Science and Technology (NTNU). It has been performed in the spring semester 2007 at the department of Telematics in collaboration with Telenor R&I.

The supervisor for this thesis has been Ivar Jørstad at UbiSafe AS and the academic responsible has been professor Do van Thanh. I would like to thank them both for help during the work with the thesis and especially Ivar Jørstad for the valuable suggestions and comments.

Trondheim, June 28, 2007

Steffen Hallsteinsen

Steffen Hallsteinsen

Contents

List of figures

List of tables

Abbreviations

AAA	Authentication, Authorization and Accounting
A8	Algorithm 8, cipher key generator
AES	Advanced Encryption Standard
AMS	Application Management System
APDU	Application Protocol Data Unit
API	Application Programming Interface
AS	Authentication Server
ATR	Answer To Reset
AuC	Authentication Center
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CGI	Common Gateway Interface
CLDC	Connected Limited Device Configuration
CSP	Credential Service Provider
EAP	Extensible Authentication Protocol
GSM	Global System for Mobile Communications
HLR	Home Location Register
HMAC	Keyed-Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMSI	International Mobile Subscriber Identity
JAD	Java Application Descriptor File
JAR	Java ARchive File
Java ME	Java Micro Edition
MAC	Message Authentication Code
MIDP	Mobile Information Device Profile
MITM	Man In The Middle
MS	Mobile Station
MSC	Mobile Switching Center

NIST	National Institute of Standards and Technology
NONCE	Number used once
OPT	One-Time Password
PDA	Personal Digital Assistant
PIN	Personal Identification Number
PPP	Point-to-Point Protocol
RA	Registration Authority
RADIUS	Remote Authentication Dial In User Service
RAND	Random number
RFC	Request For Comment
SAP	SIM access protocol
SATSA	The Security and Trust Services API
SDP	Service Discovery Protocol
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module
SMS	Short Messaging System
SP	Service Provider
TLS	Transport Layer Security
TMSI	Temporaary Mobile Subscriber Identity
UMTS	Universal Mobile Telecommunications System
UUID	Universally Unique Identifier
VLR	Visitor Location Register
WMA	Wireless Messaging API

Abstract

The number of different identities and credentials used for authentication towards services on the Internet has increased beyond the manageable. Still, the most common authentication scheme is based on usernames and passwords which are neither secure nor user-friendly. Hence, better solutions for simplified, yet secure authentication, is required in the future. This thesis present an authentication scheme based on a One-Time Password (OTP) MIDlet running on a mobile phone for unified authentication towards any type of service on the Internet.

The scheme is described in detail by an analysis and a design model. Based on the analysis and design an implementation of a prototype has been developed using Java. The security aspects of scheme are thoroughly evaluated in a security evaluation which identifies threats, security objectives and possible attacks.

The proposed solution offers a strong authentication scheme which can substitute many of the authentication schemes we are using to day. Not only can it replace the standard username/password scheme, but due to its security services it can also replace stronger schemes such as existing OTP and smartcard solutions. Therefore the solution is suitable for many services on the Internet which requires authentication such as Internet banking, corporate intranet, Internet stores and e-Government applications.

1 Introduction

1.1 Motivation

More and more services are becoming available on the Internet, and many of these services require authentication. The most widespread solution for electronic authentication today is the use of a username and password. As the number of services grows so does the number of username and password pairs that the user needs to remember. Already many people are experiencing that it is impossible to remember all the combinations and therefore they use the same pairs for all their services and choose passwords that are easy to remember. This strongly reduces the security of an already weak authentication mechanism. Several more secure solutions for electronic authentication exist, such as One-Time-Password (OTP) or Smart Card PKI solutions. They solve the security problem with passwords, but they most often increase the burden for the user. Extra hardware and software is required both for the user and service provider and they are often specific for each service so the user needs to deal with many different devices and procedures.

This thesis tries to address this problem by proposing an authentication scheme using the mobile phone as an authentication token. The GSM system is already well understood by the user and the proposed solution does not require any extra hardware device at the user side. By using the mobile phone as hardware token, strong authentication can be realized with a system and a device that the user already has and knows how to use.

The mobile phone has several advantages which can be exploited in user authentication:

– most people already have one
– the mobile phone has computing and communication capabilities that allows automation of the authentication process, relieving the user of this burden
– there are already good security mechanisms built into the GSM system that may be exploited

1.2 Problem definition

A previous project report [1] by this author has identified several solutions which utilize the mobile phone and the SIM card for authentication towards services. These solutions use different communication channels and patterns in order to securely establish the identity of the user (e.g. through SMS, through Bluetooth and Internet etc.). The project concludes with a suggestion to further study the solutions based on the OTP principle.

The goal of this thesis is to continue the work from the project and demonstrate the feasibility of the recommended solution by developing a prototype implementation and show that is works.

The solutions presented in the project report are described and analysed at a fairly abstract level including the recommended one. It must therefore be further elaborated and concretized and the justification revisited the before the analysis and design can be done.

The main tasks for this thesis are

a. Study of required background information
b. Brief study of possible authentication solutions
c. Selection of which solution to implement
d. Analysis, design and implementation of selected solution
e. A thorough evaluation of the security properties of the implemented solution

1.3 Objectives

The main objective of this thesis is to find a secure solution for user authentication using the mobile phone as the security token. There exist several solutions using the mobile phone as a security token already, but this is still a field with a lot of unexplored possibilities especially regarding improved security and user-friendliness.

Using the mobile phone for user authentication in Internet services requires the combined use of both Internet and Telecom technology. This requires a study of the basic concepts of authentication, different Internet and Telecom architectures and communication protocols used between diverse devices such as a computer and mobile phone.

In order to get an insight and overview of the possibilities the mobile phone and SIM card presents for user authentication several different authentication schemes are studied. This study leads to a selection of a solution which is further elaborated.

The selected solution is analyzed through identifying use cases, system requirements and interaction diagrams. The analysis is used in a design phase to define requisite components and class diagrams. A prototype implementing the main functionality described in the analysis and design phase is developed to prove the concept of the solution.

A thorough security evaluation of the solution identifying possible threats, security objectives and possible attacks is done to identify possible weaknesses and areas of improvement for future work and similar solutions.

1.4 Related work

Using the mobile phone as an authentication token is a topic of research which has been subject to more and more interest in the last years. A variety of solutions exist and some solutions applying the OTP principle in one way or another are becoming available on the market. This section will briefly review some of the solutions that are available today.

The Free Auth Project has made a MIDlet version of their OTP solution [2]. It is an extended version of Mobile OTP (known as mOTP+) [3] created to be used among other things as a Single Sign On client. It generates one time passwords (a hash of time+magic+pin) on Java enabled mobile phones. Their OTP generation is based on a time factor and requires that the client and server are synchronized for the solution to work. Time synchronization is not an easy task and the solution is vulnerable if the synchronization fails.

Several solutions exist where an OTP is sent to a mobile phone and used in addition with a static password or other authentication tokens. NordicEdge AB [4] uses this approach in their OTP solution. Even though additional hardware is avoided extra user operations are needed and the OTP is sent two times over the network in clear text. Several Norwegian companies such as SkandiaBanken[1] and Skatteetaten[2] are using this approach on their website to authenticate their customers.

[1] See www.skandiabanken.no
[2] See www.altinn.no

Deepnet Security offers a standalone authentication solution based on the reference architecture set forth by the initiative for open authentication (OATH) [5]. The solution offers two-way authentication and support both event-based and time-based OTP. The details of their solution are not publicly available but a high level illustration of the solution is given in Figure 1.

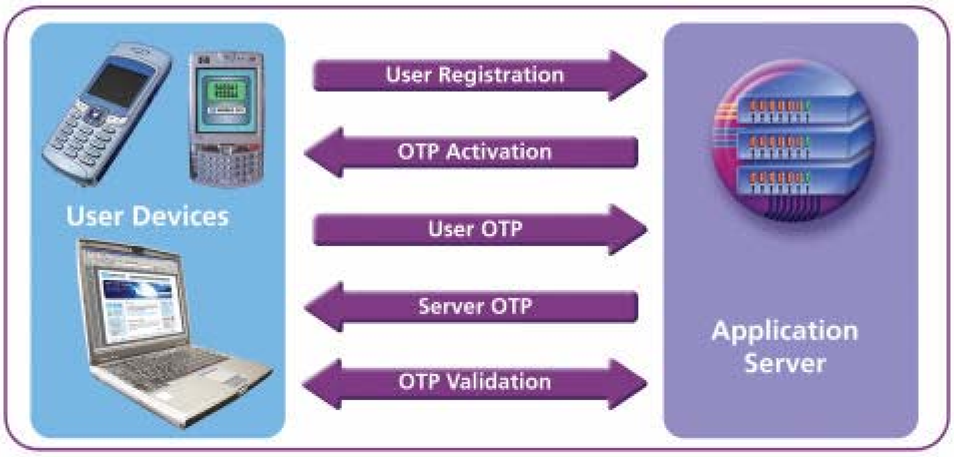


Figure 1. Deepnet Security authentication solution

Even though several solutions using the mobile phone as an OTP token exist there are still many aspects which can be studied in order to improve both the security and usability of authentication schemes. No solutions are yet to be widely used and for companies to substitute their old authentication solutions great improvements from existing solutions both for the company and its customers must be proved.

1.5 Organization of the thesis

This thesis is divided into eight chapters and an appendix. Table 1 presents a short description of each chapter.

Table 1. Organization of the thesis

Chapter	**Description**
Chapter 1 Introduction	The introduction provides a motivation for the thesis work, definition of the problem, and the main objective of the thesis.
Chapter 2 Background	The background starts with an authentication overview which presents central models, concepts and terminology in electronic authentication. Then the underlying technology used throughout the thesis is explained.
Chapter 3 Authentication using mobile phones	This chapter starts with presenting the concepts necessary for authentication using mobile phones. Then four different authentication schemes are described. The chapter is concluded with the selection of one solution which is further elaborated.
Chapter 4 Analysis	The analysis consists of requirements, use cases and interaction diagrams.
Chapter 5 Design	The design of the solution is presented through component models and class diagrams.
Chapter 6 Implementation	The implementation presents a possible deployment of the system and describes the different components of the prototype.
Chapter 7 Security Evaluation	The security evaluation presents a threat model, describes the security objectives of the solution and described possible attacks
Chapter 8 Conclusion	The conclusion summarizes the results of the thesis and presents some thoughts on future work.
Appendix A	This appendix contains an overview of and link to the code of the prototype which is available on DAIM.
Appendix B	This appendix contains a paper by this author named “Using the mobile phone as a security token for unified authentication” which is accepted for the IARIA conference ICSNC 2007.

2 Background

This chapter provides necessary background information for the understanding of the thesis. The basic concepts of authentication are introduced and the underlying technology used to build the authentication solution are presented.

2.1 Authentication overview

This section presents an overview of central models, concepts and terminology in electronic authentication. The challenge of remotely authenticating an individual person over an open network is presented, together with the most widely implemented methods for doing so. Many of the elements in this chapter are inspired by [6].

2.1.1 Authentication model

Authentication is the assurance that the communicating entity is the one that it claims to be [7]. A system can authenticate a user to determine if the user is authorized to perform an electronic transaction or get access to information or a system. Normally the authentication and transaction takes place across an open network such as the internet, but the network might also be a private network.

Authentication begins with registration. A typical registration process is shown in Figure 2. The user applies to a Registration Authority (RA) to become a subscriber of a Credential Service Provider (CSP). The subscriber is given or registers a secret, called a token, and a credential that binds the token to the username. The token and the credential are used to authenticate the user. There is always a relationship between the RA and the CSP and quite often they are separate functions of the same authority.

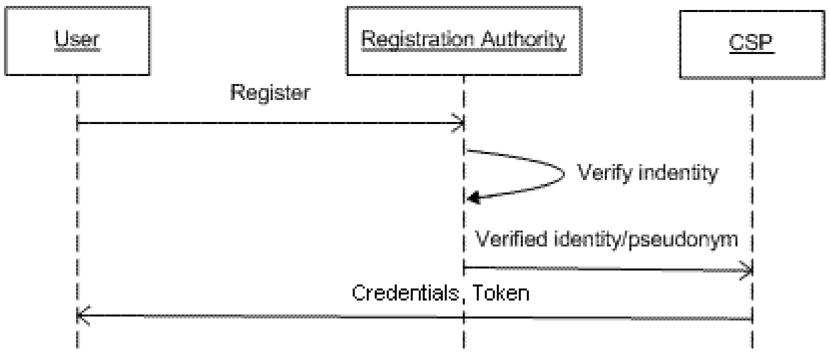


Figure 2. Registration process

The subscribers name may be either a verified name or a pseudonym. If the name is associated with the identity of a real person the name is verified. To verify a name the user must demonstrate that the identity is a real identity and that he is the person who is entitled to use that identity. This is called identity proofing and is performed by the RA. At authentication level 1 (see below), names are not verified and therefore names are always assumed to pseudonyms. At level 2 it must be specified if the name is a verified name or a pseudonym. At level 3 and 4 only verified names are allowed.

When a user needs to be authenticated he demonstrates possession and control of a token to a verifier through an authentication protocol as shown in Figure 3. The verifier can then verify that the user is the subscriber he claims to be. The verifier passes on an assertion about the identity of the subscriber to the relying party. The assertion contains identity information about the subscriber given at the registration process. If the verifier and the relying party are the same entity the assertion is implicit. The subscriber's identity can also be stored in credentials such as a public key certificate which is made available by the user. Then the relying party can use this information to authenticate the subscriber directly.

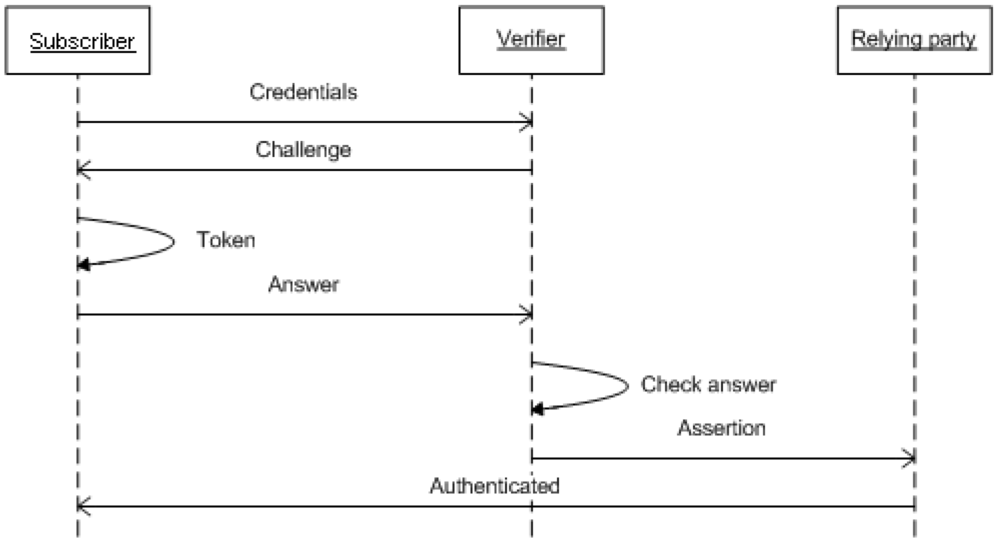


Figure 3. Authentication process

Authentication only establishes identity, and does not say anything about what the identity is authorized to do, or what access privileges it has. It is the relying party that uses an authenticated identity to make decisions about authorization and access control.

The equipment or software acting on behalf of the subscriber in the authentication process will in the remainder of this report be called the client. The verifier will be realized as an authentication server. The relaying party will be the service provider which offers the service the users wants access to.

2.1.2 Authentication levels

Authentication levels are used to categorize different authentication schemes. The levels are defined in terms of the consequences of authentication errors and misuse of credentials. The higher the consequences of misuse are the higher are the required level of assurance. In [1] four authentication levels are defined where level 1 is lowest and level 4 is highest.

Level 1 - Little or no confidence in the asserted identity’s validity

Level 2 - Some confidence in the asserted identity’s validity

Level 3 - High confidence in the asserted identity’s validity

Level 4 - Very high confidence in the asserted identity’s validity

Level 1 - There is no identity proofing at this level, but the authentication mechanism provides some assurance that the same user is accessing the data each time. It allows a wide range of authentication technologies to be used and all the token methods of the upper levels can be used. Authentication requires that the user proves to have control of the key. Passwords are not transmitted in plaintext, but cryptographic methods that block offline attacks are not required. For example, simple password challenge-response protocols are allowed.

Level 2 – Level 2 provides single factor remote network authentication. Here identity proofing is introduced, but not required. Still a wide range of authentication technologies can be used and all token methods from level 3 and 4 are allowed as well as passwords and PINs. Authentication requires that the user proves through a secure protocol that he controls the token. Eavesdropping, replay and on-line guessing attacks are prevented.

Level 3 – Level 3 provides multifactor remote network authentication. Identity proofing is required and authentication is based on proof of possession of key or one-time password through a cryptographic protocol. Level 3 authentication requires that the token is protected by a strong cryptographic mechanism. This prevents eavesdropping, replay, online guessing, verifier impersonation and man-in-the-middle attack. A minimum of two authentication factors are required. Soft token, hard token and one-time password token can be used. The user must prove that he have control of the token, and must first unlock the token with a password or biometric, or also a password in a secure authentication protocol, to establish two factor authentication.

Level 4 – Level 4 provides the highest remote network authentication assurance available. It is similar to level 3, but only hard tokens are allowed. Level 4 requires strong cryptographic authentication of all parties and all sensitive data transfers between the parties. Either public key or symmetric key technology may be used. Authentication requires that both parties prove that they have control of the token. Eavesdropping, replay, online guessing, verifier impersonation and man-in-the-middle attack are prevented.

2.1.3 Electronic credentials

Electronic credentials are the electronic world's substitute for paper credentials such as passports, driver's licenses and birth certificates. Electronic identity credentials bind a name and possibly some other attribute to a token. Given this credential and token it must be possible to verify the identity of the subscriber. Electronic credentials may be general purpose credentials or targeted to a particular verifier. There exist a lot of different electronic credential types today, and there are continuously coming new ones on the market. Some of the most common types include:

- X.509 public key identity
- X.509 attribute certificate
- Kerberos tickets that are encrypted messages

Credentials may be stored as data in a directory or database. These entities can be either trusted or untrusted depending on the credentials. Signed objects such as certificates are self-authenticating and can be stored at an untrusted entity, while unsigned credentials must be stored in a trusted database or directory that can authenticate itself to the relying party or verifier.

2.1.4 Tokens

Tokens are something the user possesses and controls that may be used to assert the user's identity. This token is a secret that should be known only to the user and therefore needs to be protected. A token may, for example, be a password that the user presents to the verifier when he needs to be authenticated.

Authentication systems can be ranked by the number of factors they use. In [1] three main factors are used:

- Something you know (password or pin)
- Something you have (ID badge or cryptographic key)
- Something you are (voice print or other biometric)

The more factors that are used in the authentication system, the stronger it is. A system that uses two or three factors are considered as strong authentication systems, while a system that uses only one token is considered a weak authentication system.

A multifactor system may be implemented so that several factors are presented to the verifier, or some factor can protect a secret. For example, a password can protect a cryptographic key stored in a hardware device. Even though only one factor is presented to the verifier this is considered a two factor authentication since two factors, the password and the device, are needed to know the secret. An impostor needs to steal the device and know the password to use the token.

The secrets are often either public key pairs or shared secrets. In a public/private key pair the private key is used as a token. The verifier, knowing the users public key by a certificate, can confirm that the user has the right private key and therefore authenticate the user. Shared secrets can be symmetric keys or passwords. These are used in the same way, but since passwords are often memorized they are considered something you know instead of something you have. They are also more vulnerable to network attacks since they have fewer possible values than keys.

2.1.5 Token types

Tokens can be divided into four kinds. Each kind uses one or more of the authentication factors mentioned earlier (something you know, something you have, something you are). Strong authentication tokens use two or more factors. The four kinds are:

Password token – is a secret that are memorized by the user and that is presented to authenticate the user's identity. Passwords are typically secret words, phrases or arbitrary character strings.

One-time-password (OTP) – is a unique secret word, phrase or character string that can be used only once. It has a limited lifetime and is useless after the limit has expired. OTPs are considered more secure than normal passwords since they are only used once and therefore can not be replayed.

Soft token – is a cryptographic key that is stored on disk or some other media. The user is authenticated by proving possession and control of the key. The soft token key is encrypted under a key derived from some activation data. This is typically a password know only to the user and is required to activate the token.

One-time-password token – is a personal hardware device that generates one-time passwords for authentication. The device can have some input pad for passwords, biometric reader or a direct computer interface. The one-time password is either displayed and typed in manually as a password or transferred directly from the token to the computer.

Hard token – is a hardware device that contains a protected cryptographic key. Authentication is provided by proving possession of the device and control of the key. A password or a biometric is required to activate the key.

2.1.6 Token levels

The different kinds of tokens satisfy different authentication levels depending on their vulnerability to different attacks. A general summary of the authentication level for tokens are given below:

- Passwords satisfy the requirements for level 1 and 2.

- Soft tokens can be used at authentication levels 1 to 3, but must be combined with a password or biometric to satisfy level 3.
- One-time passwords satisfy the requirements for level 1 to 3 and must be used with a password or biometric to achieve level 3.
- Hard tokens that are activated by a password or biometric satisfy the requirements for level 1 to 4.

Password authentication is easy to implement and familiar to users, and is therefore the most used authentication scheme. But since the password is the only thing an impostor needs to impersonate an identity it is not very strong. Passwords are memorized and can therefore not be to long or complex. That makes them vulnerable to guessing, dictionary attacks and simple exhaustion.

To misuse a hard or soft token requires the attacker to obtain two separate things: the key and a password, or the token and the ability to enter a valid biometric into the token. Therefore both soft and hard tokens are more secure than passwords. In addition a hard token is a physical object and the owner will quickly notice if it disappears. Therefore a hard token is considered more secure than a soft token.

One-time password devices are similar to hard tokens. They can be activated by a password or a biometric to provide multifactor authentication. One-time passwords do not result in generation of a shared session key and are therefore not as secure as hard tokens.

2.1.7 Authentication threats

Authentication schemes are subject to several threats. These can both be direct attacks or security flaws in the scheme.

- Tokens can be stolen or duplicated. Passwords might be guessed or intercepted and a biometric can be copied or replicated.
- An impostor can try to claim an incorrect identity by means of stolen credentials.
- The infrastructure at the RA or CSP may be compromised.
- An eavesdropper can observe the authentication protocol to try to intercept messages between CSP and verifiers to get hold of a token.

- An impostor can pose as a subscriber to try to guess a password or pose as a verifier to trick the subscriber into giving away the secret token.
- Hijackers can take over an already authenticated session to learn sensitive information or give in invalid information.

Several steps can be taken to counter these threats:

- Multiple factor authentication makes it harder to compromise the tokens.
- Limited tries and complex passwords make them to hard to guess.
- Network communication can be encrypted to prevent eavesdropping.
- Mutual authentication makes man-in-the middle attacks hard to accomplish.

2.1.8 Authentication challenges

As mentioned before password authentication is the most common authentication scheme today. It is easy to implement and was considered very user-friendly, but as the demand for username/password-pairs increase, the user-friendliness drops drastically. Password is also the weakest kind of security token and has several serious security flaws.

Single sign-on solutions have been proposed to solve the problem with plurality, but these are not yet in widespread use and do not solve the security problems. Even though it is an improvement this is not a solution for the future since it does not deal with the security weaknesses.

There exist several stronger authentication schemes that implement the steps proposed above and more. These solve many of the security problems associated with password tokens, but suffer from other significant drawbacks as pointed out in [8].

- They are expensive both for users and providers since extra hardware devices and dedicated software are required at the user side and providers need to support specific API and handshake protocols for each device.
- Their lack of interoperability makes it impossible for them to operate with each other.
- They do not scale well which makes it difficult to extend them to global solutions.

As we see the need for secure user-friendly authentication schemes is strong. In order to work efficiently these authentication schemes need to be designed with great consideration for human strengths and limits, in addition to the technological challenges.

2.2 Cryptography

This section presents some basic cryptographic concepts used throughout the thesis.

2.2.1 Web Security

There are several ways to obtain web security today. The most popular choice to secure web applications is to use the secure Hypertext Transport Protocol (HTTPS). HTTPS is not a separate protocol but refers to a combination of a normal HTTP interaction over an encrypted Secure Socket Layer (SSL) transport mechanism

SSL was developed by Netscape as a protocol for providing a reliable end-to-end security service over TCP. The SSL Record Protocol provides basic security services to higher-layer protocols such as the HTTP.

SSL uses public cryptography such as RSA to exchange session keys for encrypting and authenticating a session. When the session keys are exchanged symmetric encryption is used to secure the link. Message integrity is ensured by using a Message Authentication Code (MAC) also based on a session key.

To exchange the session keys SSL uses a handshake protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and session keys to be used to protect the session. The protocol consists of a series of messages exchanged by the client and server as shown in Figure 4 [7].

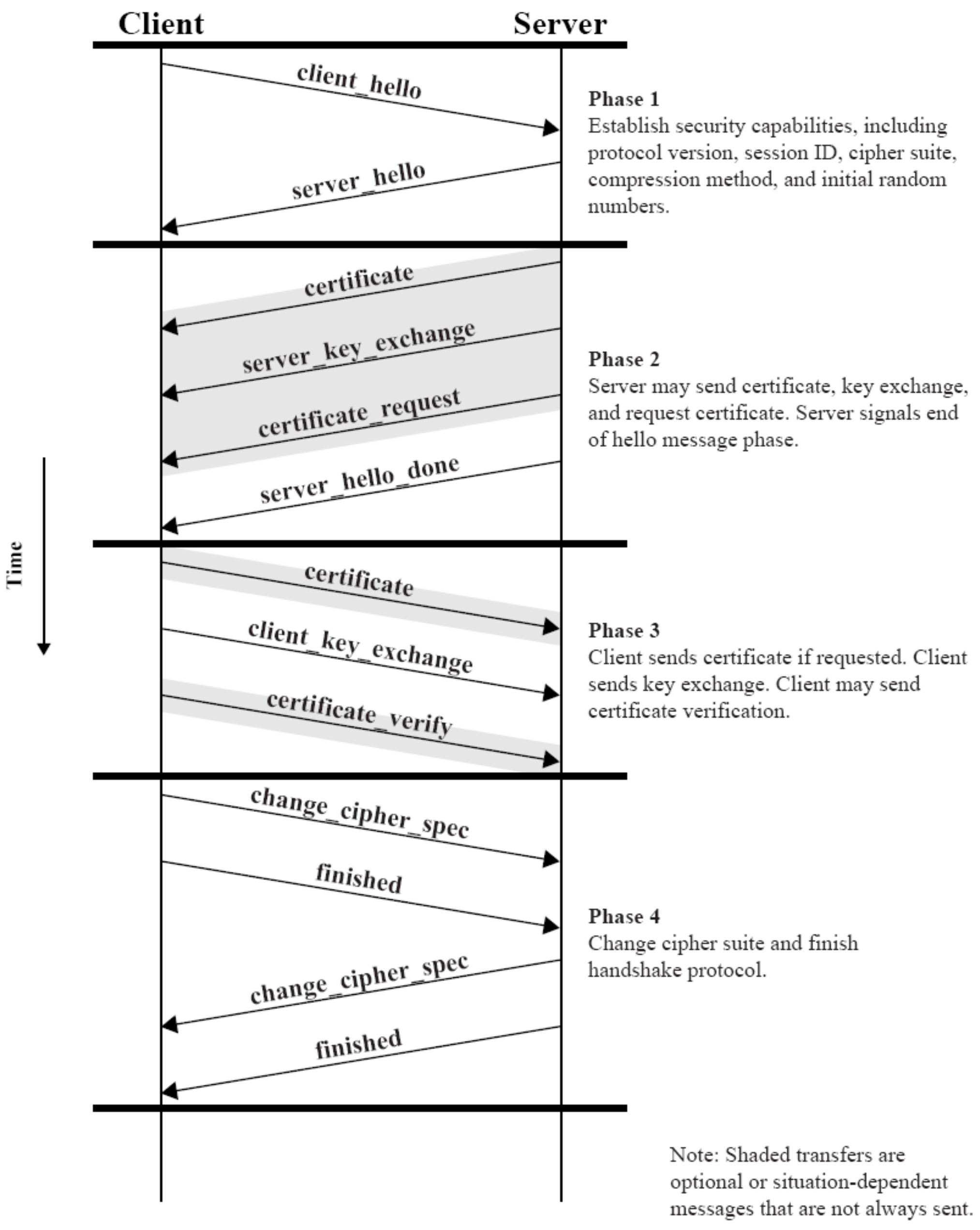

Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Figure 4. SSL Handshake Protocol

Today the term Transport Layer Security (TLS) is used just as often as SSL. TLS is the IETF standard of SSL and the first version on TLS can be seen as SSLv3.1 since there were very few differences from SSLv3.

2.2.2 One-Time Password

One-time passwords (OTP) solve many of the problems with human-made passwords. They are generated by a computer and can therefore appear to be totally random. In addition as the name suggests each password is only used once. That makes them invulnerable to replay attack and sniffing. By changing each time a password is needed OTP solutions introduce liveness. This is an important concept in security since it is possible to detect if someone is using old information.

There exist three different types of OTP. The first type uses a mathematical algorithm to generate a new OTP from a previous one. The second type is based on time-synchronization between the authentication server and the client. The third type is based on a challenge, e.g. a random number that is fed into a one way function.

2.2.3 Secure hash functions

The security of the OTP system is based on the non-invertability of a secure hash function. Such a function must be relatively easy to compute in the forward direction, but computationally infeasible to invert [9]. A hash function takes a message M of any length and produces an fixed-size message digest H(M) as output.

A hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is referred to as the one-way property.
5. For any given block x, it is computationally infeasible to find y ≠ x with H(y) = H(x). This is referred to as weak collision detection.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is referred to as strong collision resistance.

SHA-1 Secure Hash Function

One of the most used hash functions and the one used in this thesis is the Secure Hash Algorithm (SHA). It was developed by the National Institute of Standards and Technology (NIST) in 1993. A revised version was published in 1995 and is generally referred to as SHA-1. The algorithm takes as input a message with a maximum length of 2^{64} bits and produces as output a 160-bit message-digest [10].

2.2.4 Message Authentication Code

A Message Authentication Code (MAC) is used to protect messages from active attack such as falsification of data and transactions. The use of a MAC is most convenient when it is important to ensure that the sender and content of the message is authentic while hiding the content from others is not necessary. Since encryption increases the resource use there are sometimes desirable to authenticate a message without encrypting it. This can be done by a MAC. A MAC is a small block of data computed as a function of the message and encrypted with a key shared between the sender and receiver. The block is appended to the message before it is sent. The recipient performs the same calculation on the received message, using the same secret key, to compute a new MAC. The received code is compared to the calculated code and if they match the receiver knows that the message is authentic. The usage of a MAC is illustrated in Figure 5.

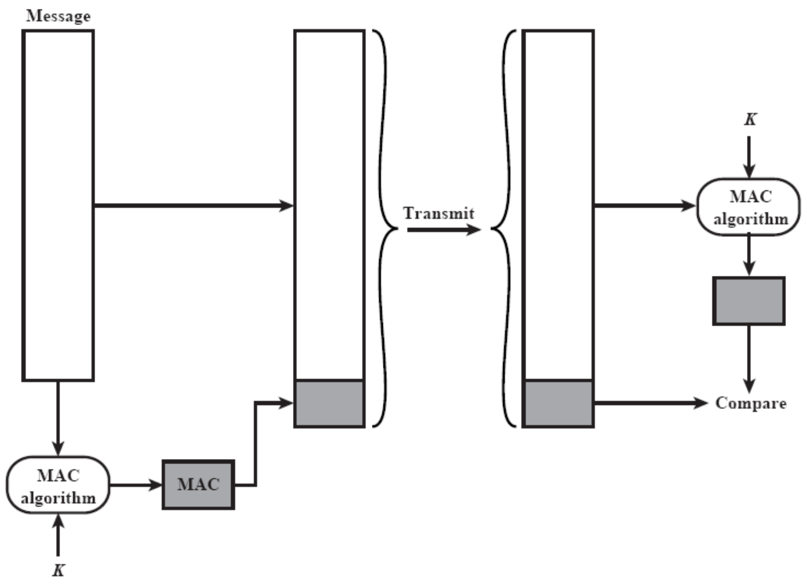


Figure 5. Message Authentication Code

HMAC

HMAC was developed as a result of the increased interest in a MAC derived from a cryptographic hash code such as SHA-1. HMAC uses a secret key together with a secure hash function to produce a 160 bit hash of a message which can be used as a MAC [11].

2.2.5 Advanced Encryption Standard

Advanced Encryption Standard (AES) is one of the most popular algorithms for symmetric encryption today. It was designed by Daemen and Rijmen in1998 and was chosen to be the new encryption standard for the U.S. Government in 2002. AES encrypts 128 bit blocks and can be used with key lengths from 128 to 256 bit[12].

2.3 GSM

This part starts with a short presentation of the GSM network architecture and the most important elements. Afterwards follows a more detailed description of the authentication scheme used in GSM.

2.3.1 Architecture

GSM is the most widely deployed and used telephone system in the world today. It is a digital system and it is the base for newer systems such as UMTS. The main elements are specified below.

The handset, called mobile station (MS) is made out of two parts - the mobile phone and the Subscriber Identity Module (SIM). The SIM is a smartcard containing subscriber identity, authentication information and service information. This information is basically contained in two information items.
The International Mobile Subscriber Identity (IMSI) is a unique number for each subscriber in the world. This is the identity of the subscriber in the network and contains the information about the home network and the country of issue.
The Ki is the root encryption key. This is a randomly generated 128 bit number allocated to a particular subscriber and seeds all the keys and challenges in GSM.

As will be shown the SIM plays a crucial part in the authentication scheme. It is only when the SIM is inserted into the phone that the subscribers services becomes available.

The MS communicates over the air with a Base Transceiver Station (BTS). One or more BTSs are connected to the Base Station Controller (BSC) and these two make up the Base Station Subsystem (BSS). The BSS provides the radio interface to the MS and handles radio resource management and mobility management.

The BSS is connected to a Mobile Switching Center (MSC) that controls call setup, call routing and other switching tasks. The Visitor Location Register (VLR) is also found here.

This is a database that contains subscriber-related information during the time period that the MS is in the coverage area of the MSC. The MSC is connected to other networks through a gateway.

The Home Location Register (HLR) contains subscriber data, such as service details. In association with the HLR we find the Authentication Center (AuC). This is a network element that contains subscriber-specific authentication data such as the authentication key and one or more authentication algorithms. More details can be found in [13]. The GSM architecture is shown in Figure 6 [14].

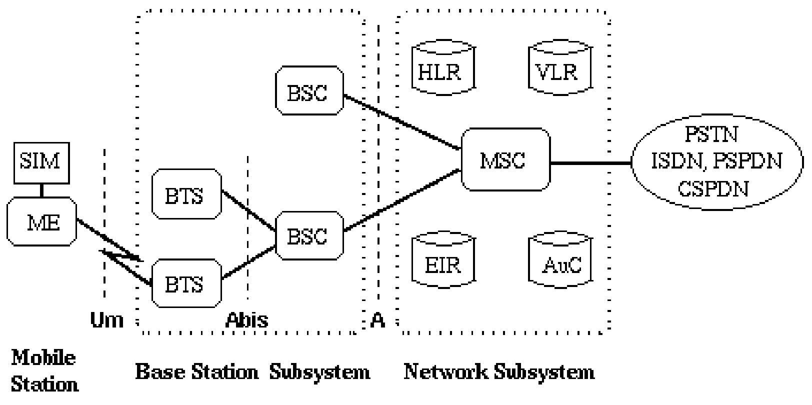


Figure 6. GSM Network Architecture

2.3.2 Authentication

User authentication is very important in GSM since users are mobile and change their point of attachment to the network quite often. Authentication makes sure that only authorized users are allowed access to the network and that the bill goes to the right user. It also enables secure communication between the MS and the network by creating an encryption key. The authentication procedure in GSM is described thoroughly in [15].

Two steps of authentication are used in GSM. First the human user is authenticated with the phone to ensure that the phone is used by the real owner. Second the phone is authenticated to the network to ensure that only authorized subscribers can log on to the network. The solution chosen in GSM to implement this is the use of the SIM card. The SIM card is issued to the user by a service provider and is protected by a password, a four digit Personal Identity Number (PIN). When the SIM is inserted into the phone and the phone is turned on the PIN must be entered before the user gets access to all the features of the phone. This is a local authentication and nothing is transmitted over the air. This authentication is performed by the

SIM and the phone only acts as an interface between the user and the SIM. Next the SIM must authenticate itself to the network. As long as the phone is on this happens on a regular basis without any involvement from the user.

The authentication of the SIM by the network is based on a challenge/response protocol and a shared secret. The shared secret is a symmetric key (Ki) stored in the SIM and at the AuC in the subscriber's home network.

When a SIM is to be authenticated the AuC computes a value (SRES) by feeding the Ki and a random number into an Algorithm called A3. This is illustrated in Figure 7 [16].

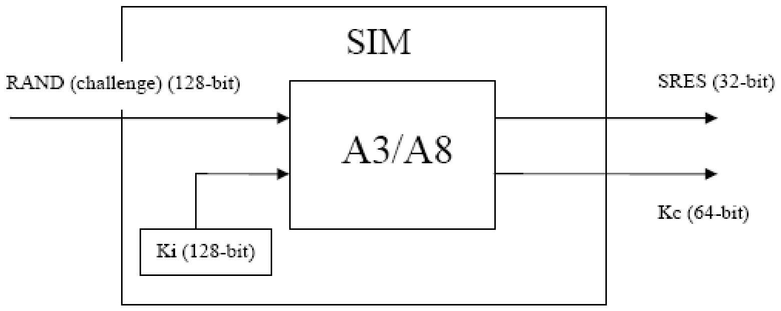


Figure 7. Generation of SRES and Kc

The random number is sent over the air to the MS and the SIM performs the same operation. Then the SRES computed by the SIM is sent back and is compared with the SRES computed by the network. If they match the SIM is authenticated. Since the SIM is the only one with knowledge of the key no one else can compute a correct SRES. The roles and steps of the GSM authentication scheme is illustrated in Figure 8 [16].

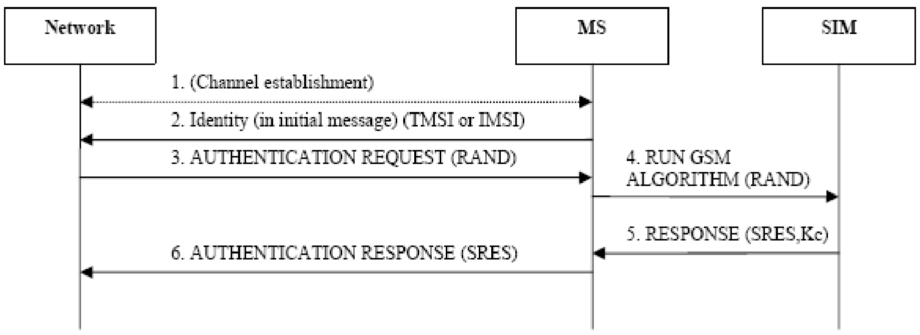

Network
MS
SIM
1. (Channel establishment)
2. Identity (in initial message) (TMSI or IMSI)
3. AUTHENTICATION REQUEST (RAND)
4. RUN GSM ALGORITHM (RAND)
5. RESPONSE (SRES,Kc)
6. AUTHENTICATION RESPONSE (SRES)

Figure 8. GSM authentication scheme

The main security goal in GSM is confidentiality. This is obtained by encryption of the link between the MS and BTS. As mentioned earlier the encryption is based on the authentication. The reason is that the encryption key (Kc) is generated in the same process as the SRES as shown in Figure 7.

This is an effective way of doing it, but it means that encryption can not start before the SIM is authenticated. The problem with this is that the MS has to send the IMSI in clear text over the air. This is a potential security threat since it opens for theft of the IMSI and other privacy issues. The GSM solves this by using a Temporary Mobile Station Identifier (TMSI). This is assigned to the MS after the first logon to the network and is the identity used afterwards.
The TMSI is only valid in one coverage area and is therefore replaced when the MS moves to another area. This is handled by the VLR. The MS stores the TMSI on the SIM card and can therefore use the TMSI also after it has been switched of for a period. The use of TMSI makes it only necessary to send the IMSI over the air once, which makes the probability for someone stealing it very small. It can be required later if the mapping between the IMSI and the TMSI is lost.

Since authentication can happen quite often in GSM this could potentially lead to a lot of signaling traffic between the serving MSC and AuC in the home network. To avoid the signaling taking a long time going from the AuC to the MSC when the MS is roaming, the serving MSC authenticates the MS directly. Since the MSC can not have knowledge of the Ki some information with the AuC has to be exchanged. The solution is that the AuC precomputes authentication triplets that are stored at the MSC. These triplets contain a random number, the associated SRES and the resulting Kc. With this information available the MSC can authenticate the MS without knowledge of the Ki. The triples are sent to the MSC in batches and therefore the MSC only has to contact the AuC once in a while to update the triplets.

2.3.3 Security issues

When GSM was designed the security requirements were that is was supposed to be as good as that of the wireline systems. This was achieved, bus since the wireline systems had no security mechanism the security could have been better. Today several flaws in GSM security have been revealed [16].

The network does not authenticate itself to the phone. This is the most serious flaw in the GSM system. The authentication scheme described above does not require that that the network shows any knowledge of the Ki, or any other authentication context to the phone. This opens for a man-in-the-middle attack where the attacker can set up a fake base station and listen to all the traffic going through it. This is improved by implementing 2-way (mutual) authentication in UMTS. 2-way authentication can also be obtained by the use of the EAP SIM protocol (see next chapter).

Several weaknesses have been found in the common implementation of the A3 and A8 algorithm. Many providers have changed to newer implementations of these algorithms that are considered more secure. In UMTS totally new algorithms are implemented.

Key length is only 64 bits. Today a minimum of 128 bit is required for keys to be considered safe. 128 bit keys are used in UMTS. The GSM keys can be strengthened by combining several keys to get longer key lengths. This is done in EAP-SIM.

If the mapping between the IMSI and TMSI is somehow lost the MS is asked to send its IMSI instead. This happens in clear text since the MS is not yet authenticated. This can be used by an attacker to get the mapping between the TMSI and the IMSI since both is sent over the air.

Replay attack is not prevented in GSM. Both triplets and authentication messages can be replayed. UMTS prevents this by using sequence numbers.

2.3.4 GSM Short Message System (SMS)

SMS makes it possible to send short text messages over the GSM network. The message length is limited to 160 alphanumeric characters and can not contain either images or graphic. The main features for this service is speed, cheap rates and the guarantee that the message will reach the receiver, even though the phone is out of radio coverage or turned of.

When a message is sent it is handled by a Short Message Service Center (SMSC) which relays it to the appropriate receiver. The SMSC contacts the HRL to find the location and status of the receiver. If the receiver is inactive or out of range the SMSC stores the message. When the receiver becomes active or gets coverage the HLR notifies the SMSC which attempts to deliver the message. The message is transferred by a SMS message delivery system in a Short Message Delivery Point-to-Point format. The system pages the receiver and, if it responds, delivers the message. The SMSC is notified that the message is received and marks the message as sent so it will not be sent again. The SMS delivery mechanism is illustrated in Figure 9 [17].

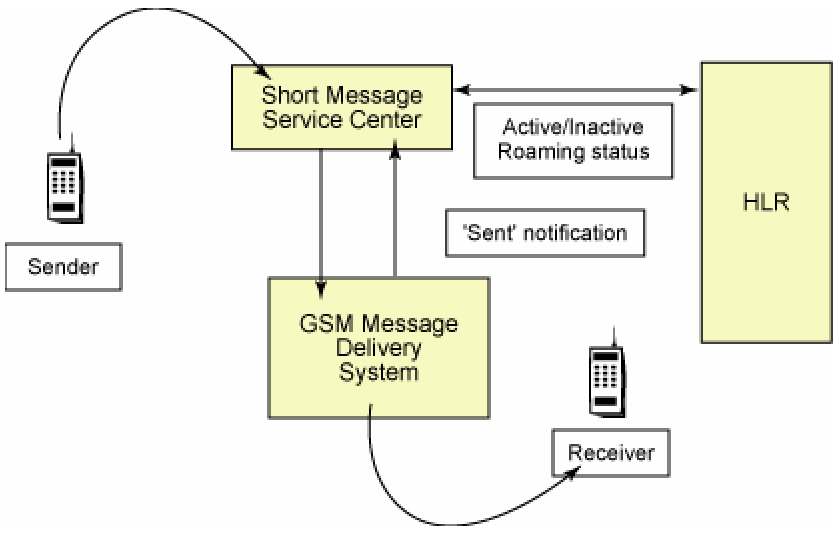

Short Message Service Center
Active/Inactive Roaming status
HLR
'Sent' notification
Sender
GSM Message Delivery System
Receiver

Figure 9. SMS Delivery System

2.4 Extensible Authentication Protocol (EAP)

As discussed in the previous part there are some security considerations with the GSM system. One way to improve the security in GSM is to use it together with the Extensible Authentication Protocol (EAP). By doing this both 2-way authentication and stronger end-to-end security can be realized. This chapter presents the Extensible Authentication Protocol (EAP) which is defined in the RFC3748 [18] and the specification for use in mobile environments defined in RFC4186 EAP-SIM [19].

EAP is an authentication framework which supports multiple authentication methods. It typically runs directly over data link layers such as PPP or IEEE 802, without requiring IP.
EAP was originally intended to be used over PPP, but has also been adopted by the IEEE 802.11i standard.

EAP has a set of messages that is used to initiate and end an authentication procedure. In a way EAP can be seen as an agent. It introduces the communicating parties to each other, then they choose their preferred authentication protocol and at the end EAP closes the deal.

2.4.1 EAP Messages

EAP uses four types of messages. These are:

- Request: Messages sent from the authenticator to the supplicant
- Response: Messages sent from the supplicant to the authenticator
- Success: Sent from authenticator when access is granted
- Failure: Sent from authenticator when access is denied

The messages sent from the authenticator can also be sent from an authentication server when this is used.

Request and response messages are further divided using EAP Type fields. The Type fields are used to indicate what information is being carried in the EAP messages. The most important Type field is the Identity field. This is used to establish the identity of the supplicant. Other type fields can be used to send user-displayable text or authentication protocol specific messages.

Because of the use of the Identity field EAP can be used as a self contained authentication protocol. This means that when EAP is used together with other protocols several authentication methods are used in sequence. This concept is called serial authentication and allows you to run as many authentication methods in sequence as you wish before the final EAP-Success or EAP- Failure message. This can be exploited in new approaches that allow the client to authenticate the network before revealing its identity [20].

2.4.2 EAP message exchange

EAP defines the following message exchange sequence to authenticate a supplicant. Figure 10 gives an illustration.

1. The authenticator sends a Request to authenticate the supplicant with the Type field indicating what is being requested. This will typically be the identity and the use of

some authentication method, for example MD5-challenge. The identity request might be bypassed if the identity is implicit or obtained in some other fashion.

2. The supplicant answers with a Response packet containing the same Type field as in the Request.

3. The authenticator sends another Request packet and the supplicant replies with a Response. The sequence of Request and Responses continues as long as needed. The EAP protocol is a ‘lock step’ protocol so a new Request can not be sent before receiving a valid Response.

4. The message exchange continues until the authenticator cannot authenticate the supplicant, and ends the conversation with an EAP-Failure message, or the authentication is successful and the authenticator sends an EAP-Success message to the supplicant and finishes the authentication.

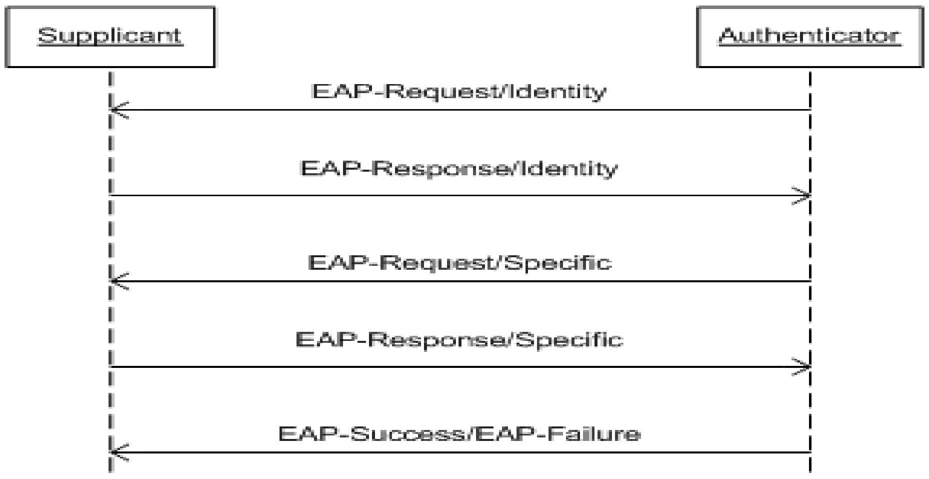


Figure 10. EAP message Exchange

2.4.3 EAP-SIM

EAP-SIM specifies the use of the EAP mechanism for authentication and session key distribution using GSM-SIM. The EAP-SIM mechanism introduces security enhancements to GSM authentication and key agreement by allowing use of multiple authentication triplets to strengthen the session keys. It also introduces authentication of the network so mutual authentication can be obtained. Through this the EAP-SIM improves some of the biggest weaknesses in the GSM-SIM authentication mechanism.

In GSM-SIM the session key is only 64 bits long. By today's standard, the key needs to be at least 128 bit long. To achieve this, the EAP-SIM allows two or three Rand challenges to be combined. Each time a challenge arrives the SIM card computes another 64 bit session key. These session keys are joined together to produce a session key of arbitrary length. To avoid that the attacker just have to crack two or more independent 64bit keys, the keys are not simply concatenated, but combined with a NONCE from the SIM using a hash algorithm.

Anonymity in GSM is provided with the use of a TMSI. EAP-SIM obtains user anonymity by introducing IMSI privacy. During authentication the server and the MS agree on a new subscriber identity to be used in the next authentication. This is called a pseudonym. This pseudonym is set using encryption and is changed every time the device connects to the network. This makes it useless to observe a large number of authentications since an attacker can not know which pseudonym belongs to which device.

The fact that the device can not authenticate the network is a big drawback in GSM. The MS has no possibility to confirm that it is connected to a legitimate network. This problem is resolved in EAP-SIM by having the MS send a nonce value at the start of the authentication. The network has to incorporate the nonce value into an encrypted response, and to do this correctly it needs access to legitimate triplets. Since both sides need access to legitimate triplets to be able to produce matching SRES values mutual authentication is obtained.

Figure 11 shows an example of full EAP-SIM authentication.

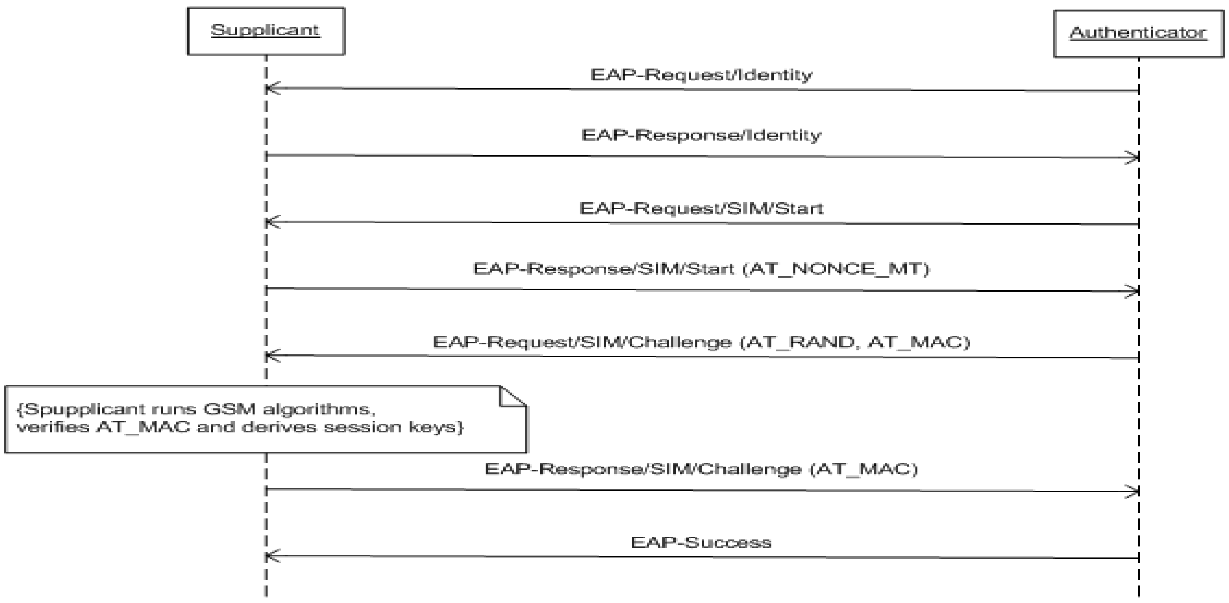

Supplicant
Authenticator
EAP-Request/Identity
EAP-Response/Identity
EAP-Request/SIM/Start
EAP-Response/SIM/Start (AT_NONCE_MT)
EAP-Request/SIM/Challenge (AT_RAND, AT_MAC)
{Spupplicant runs GSM algorithms, verifies AT_MAC and derives session keys}
EAP-Response/SIM/Challenge (AT_MAC)
EAP-Success

Figure 11. Full EAP-SIM authentication

In some environments EAP authentication may be required quite frequently. To use the full EAP-SIM authentication requires the use of the A3/A8 algorithms and two or three new triplets from the authentication center. This is a time consuming process and is therefore not suited for these situations. EAP-SIM includes a more inexpensive fast re-authentication that does not depend on the A3/A8 algorithms. Fast re-authentication only depends on the keys derived in the preceding full authentication and an unsigned 16-bit counter and is therefore much faster than the full scheme. Fast re-authentication is optional to implement and either of the sides can fall back to full authentication at any time.

2.5 Bluetooth

Bluetooth is a short-range radio technology that is designed to provide wireless communication between different personal devices. It has become the de facto standard for such communication and is being implemented on most of today's mobile phones, PDA's and laptops. Bluetooth was developed in the mid-1990s by Ericsson Mobile Communications, but soon became an international standard supported by many companies and in 1998 the Bluetooth Special Interest Group (SIG) was founded [21].

In order for two devices to communicate over Bluetooth a Bluetooth connection has to be set up. This starts with one of the devices assuming the role of "master" and starts searching for other Bluetooth devices in range by sending out an inquiry. The devices responding to the inquiry becomes "slaves". A master can handle up to 7 slaves at the same time and such a group is called a piconet. Several piconets can be connected and form a scatternet. In the connection procedure the devices will exchange device names, synchronisation data, and other Bluetooth information.

To setup a secure link between two the devices a pairing process can be performed. The pairing requires that the two devices share a secret passkey. This is typically typed into both devices by the user. By the use of such a passkey the two devices can cryptographically authenticate each other and also exchange encryption keys for encrypting the link.

In Figure 12 the Bluetooth protocol stack is illustrated. The upper layers such as RFCOMM and OBEX provides an interface to higher level protocols such as point to point protocol

(PPP) and TCP/IP which can be used for communication between higher level applications running on the mobile phone and computer. The Service Discovery Protocol (SDP) defines features on the Bluetooth device as services. This enables the Bluetooth devices to discover and use the services available on the connected devices.

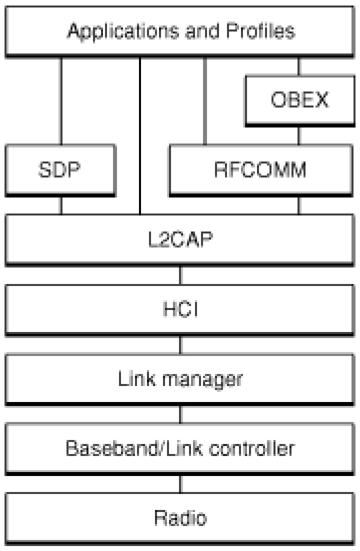


Figure 12. Bluetooth protocol stack

The Bluetooth specification also specifies several profiles supported by the RFCOMM and OBEX layer. For this thesis the SIM access profile is the most interesting.

More details about the Bluetooth wireless technology is found in [22].

2.5.1 SIM Access Profile

The SIM Access Profile (SAP) in Bluetooth is described in [23]. SAP defines procedures and protocols for handling access to a remote SIM over a Bluetooth connection. It provides a transport and control solution for GSM 11.11 [24] and GSM 11.14 [24]. The protocol stack is illustrated in Figure 13.

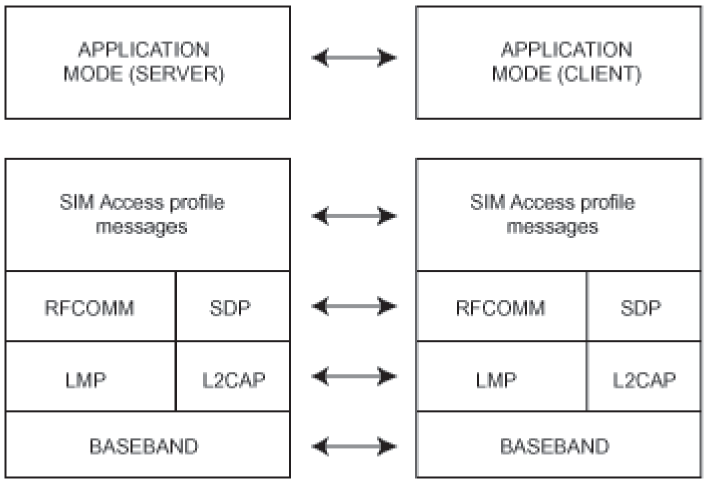

APPLICATION MODE (SERVER)
APPLICATION MODE (CLIENT)
SIM Access profile messages
SIM Access profile messages
RFCOMM
SDP
RFCOMM
SDP
LMP
L2CAP
LMP
L2CAP
BASEBAND
BASEBAND

Figure 13. SAP protocol stack

The profile can function in two different roles:

- SIM access client
- SIM access server

The client is the device that remotely accesses the SIM card and the server is the device containing the SIM.

The client can initiate the following operations to the server:
Manage connection. Establish and terminate a SIM Access Profile connection.
Transfer APDUs (Application Protocol Data Units). The application sends command APDUs, while the SIM sends response APDUs. Command and response APDUs only occur as pairs, in which each command APDU is followed by a response APDU.
Transfer ATRs (Answer to Reset). Sends the ATR content from the server to the client over the Bluetooth link. ATRs are used to report that the receiving side has accepted the APDU.
Control the SIM. Turn the SIM card on or off.

The server initiates the following operations to the client:
Report status. Server informs the client about their connection status.
Transfer card reader status. Sends the card reader status to the client over the Bluetooth link [25].

2.6 Java Platform, Micro Edition (Java ME)

Java ME is a collection of API's for development of Java applications on low resource devices such as mobile phones, PDA's and other similar appliances. The collection of technologies and specifications included in Java ME can be combined to make it possible for developers to construct a complete Java runtime environment that will fit the requirements of a particular device or market.

Java ME is divided in three parts; configurations, profiles and optional packages.

Configurations are specifications that detail a virtual machine and a set of class libraries which provide the necessary APIs that can be used with a certain class of device. They provide the base functionality for a particular range of devices that share similar characteristics.

- The Connected Device Configuration (CDC) is a framework for using Java technology to build and deliver applications that can be shared across a range of network-connected consumer and embedded devices, including smart communicators, high-end personal digital assistants (PDAs), and set-top boxes

- The Connected Limited Device Configuration (CLDC) defines the base set of application programming interfaces and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream personal digital assistants.

Profiles complement a configuration by adding more specific APIs to make a complete runtime environment for running applications in a specific device category. The most common profile is the Mobile Information Device Profile (MIDP). This in combination with CLDC is a widely adopted and used on almost all Java enabled phones.

Optional packages extend the Java ME platform by adding functionality to the technology stack[26].

2.6.1 SATSA (JSR 177)

The Security and Trust Services API (SATSA) profile extends the security features for the Java ME platform by introducing API's for cryptography, digital signature and user credential management. It also defines methods for communicating with a smart card such as the SIM. By implementing SATSA a Java MIDlet running on the mobile phone can communicate with security applications implemented on the SIM Card.

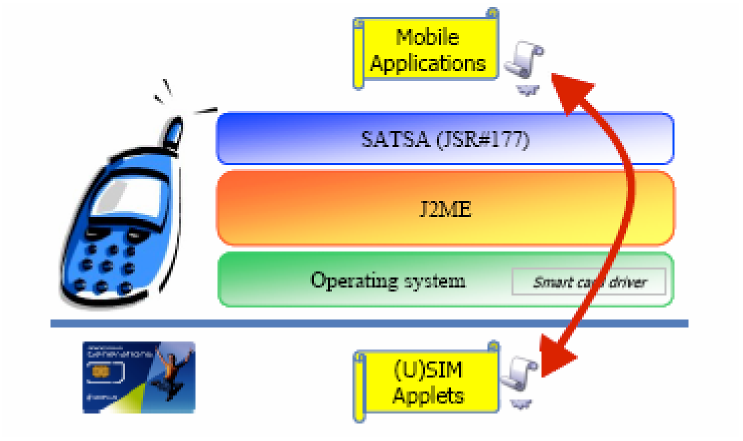


Figure 14. SATSA SIM communication

Figure 14 illustrates how SATSA run on top of the Java ME platform and enables the Java MIDlet to communicate with the applet running on the SIM.

The SATSA specification consists of four different APIs.

- SATSA-APDU: Communication with smart card using application protocol data units.
- SATSA-JCRMI: Communication with smart card using a remote object protocol.
- SATSA-PKI: Support for digitally signing of data and certificates.
- SATSA-CRYPTO: General-purpose cryptographic API that supports message digests, digital signatures, and ciphers. More about JSR 177 can be found in [27].

2.6.2 Wireless Messaging API (JSR 120)

The Wireless Messaging API (WMA) is an optional package for Java ME enabling mobile devices to communicate over a wireless interface such as SMS. WMA is based on the Generic Connection Framework and is intended for the Connected Limited Device Configuration. All the WMA components are contained in one package as shown in Figure 15 [28].

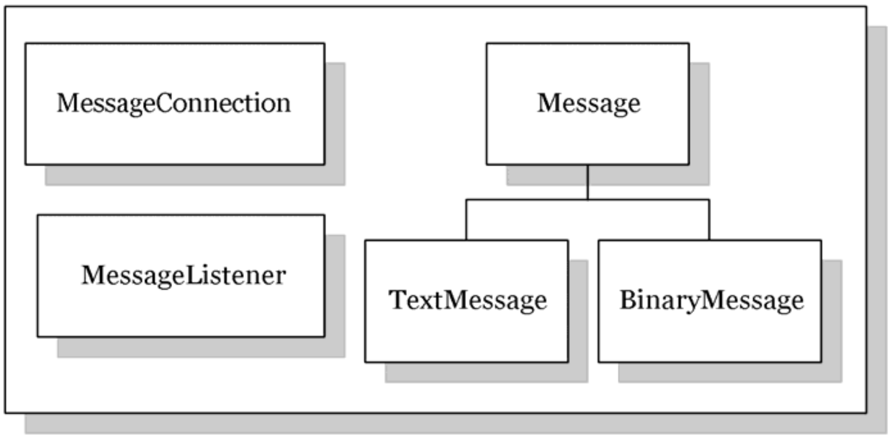


Figure 15. Components of WMA

2.6.3 Push Registry

Push Registry is a mechanism included in MIDP 2.0 which enables MIDlets to be launched automatically. The push registry manages network- and timer-initiated MIDlet activation which enables an inbound network connection or a timer-based alarm to wake a MIDlet up[29].

The push registry is part of the application management system (AMS), which is the software responsible for each application's life-cycle (installation, activation, execution, and removal). The push registry is the component of the AMS that exposes the push API and keeps track of push registrations. Figure 16 summarizes the elements of the push registry.

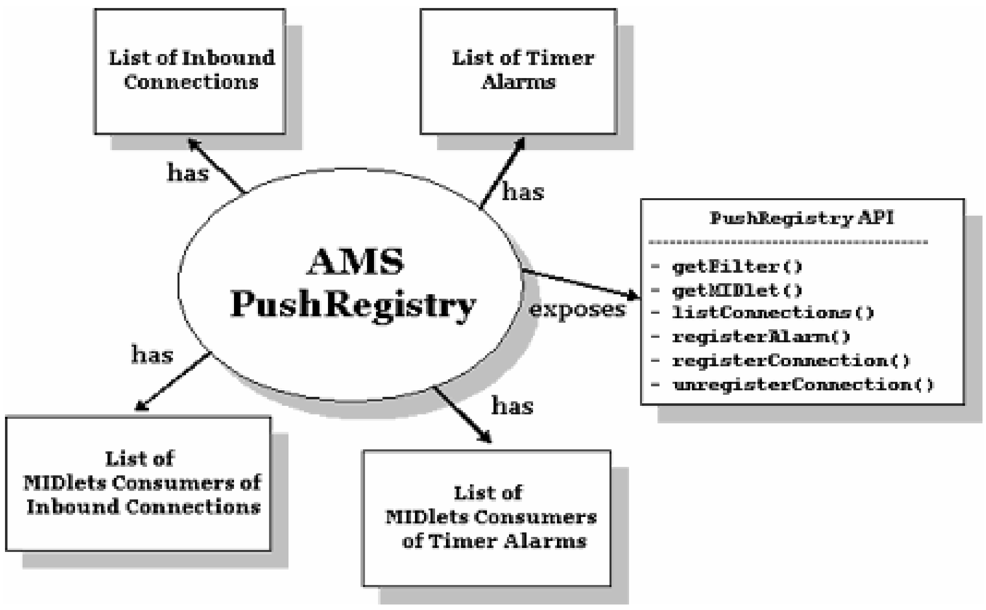


Figure 16. Push Registry elements

The push registry's behavior can be described in three steps:

1. The MIDlet is registered with a port and a protocol such that, if any message arrives in the specified port with the protocol mentioned, the application management software (AMS) on the mobile phone delivers it to the MIDlet. The registration of the port is done statically using the Java ME application descriptor (JAD) file.

2. The server sends a message to the specific device using the particular protocol and port where the MIDlet application is registered to listen.

3. After the message is delivered to the device, the AMS calls the MIDlet application, which has registered to listen to that particular port and particular protocol. Once the message is delivered to the MIDlet, it is the application's responsibility to process the message accordingly [30].

Figure 17 shows how a MIDlet is activated through a network connection.

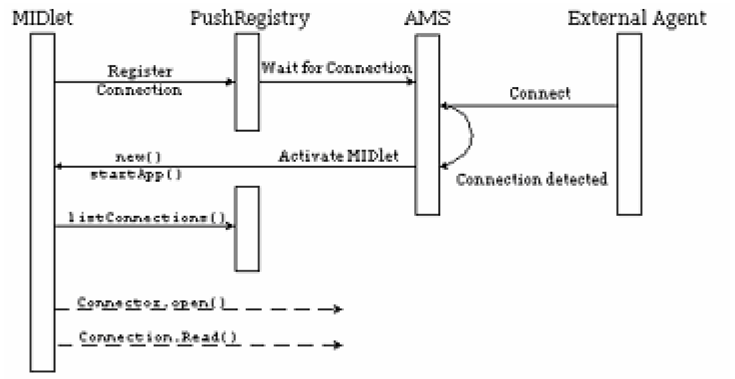


Figure 17. Push Registry network activation

2.6.4 Bluetooth (JSR 82)

JSR 82 specifies a set of Java APIs for Bluetooth communications. It allows Java-enabled devices to integrate into a Bluetooth environment[31]. The specification consists of two packages:

– javax.bluetooth: The core Bluetooth API
– javax.obex: The object exchange API

The javax.blueooth package provides an API for device discovery and service discovery. It also makes it possible to set up L2CAP and RFCOMM connections.

2.7 SIM Application Toolkit (SAT)

The SIM Application Toolkit consists of a set of commands programmed into the SIM card which define how the SIM should interact with the outside world. This enables applications on the SIM card to be presented in an easy way to the user. Applications can be entirely defined by the operator and additional Menus installed on the handset by the SIM [32].

SAT is dependent of an interface that ensures communication between the SIM and mobile phone initiated by the SIM. This is specified in GSM 11.14[24]. GSM 11.14 describes several mechanisms for SIM initiated communication:

Profile Download

Profile downloading provides a mechanism for the ME to tell the SIM what it is capable of

Proactive SIM

Proactive SIM gives a mechanism where by the SIM can initiate actions to be taken by the ME. These are:

- display text from the SIM to the ME;
- send a short message;
- set up a voice call to a number held by the SIM;
- set up a data call to a number and bearer capabilities held by the SIM;
- send a SS control or USSD string;
- play tone in earpiece;
- initiate a dialogue with the user;
- provide local information from the ME to the SIM.

Data download to SIM

Data downloading to the SIM uses the transport mechanisms of SMS point-to-point and Cell Broadcast.

Menu selection

The menu selection mechanism is used to transfer the SIM application menu item which has been selected by the user to the SIM.

Call control by SIM

This gives the SIM the possibility to control calls made from the mobile phone.

These mechanisms are also available to the SIM access profile in Bluetooth. SAT is supported by all mobile phones produced since 2000.

3 Authentication using mobile phones

This chapter describes the concepts necessary for using the mobile phone as an authentication token. First a general architecture which relies on the concept of a closed loop is presented. Then four different authentication schemes using the mobile phone and SIM card for authentication are presented. These schemes are taken from [1] and are included for the completeness of the thesis and for reevaluating which scheme will be further elaborated in the remainder of the thesis. The evaluation is found at the end of the chapter.

3.1 Introduction

Two-factor authentication is becoming more and more popular in applications that require strong authentication. Many of today’s solutions implementing two factor authentication offer better security for the users, but usually also demands higher level of knowledge and effort on the user side. The schemes presented in this chapter will use the mobile phone as a security token to obtain two-factor authentication without requiring the user to have extra hardware or knowledge.

As described earlier two-factor authentication means that two separate authentication factors, “something you know” and “something you have”, are used in the authentication. The SIM card used together with a mobile phone incorporates both things. The SIM card represents something you have, and to be able to make it work it has to be unlocked by a PIN only known to the owner, something you know. As shown a hardware token protected by a password can satisfy the requirements for the highest authentication level (level 4). Adding that the SIM card already has a built in authentication protocol which can be exploited the mobile phone with its SIM card represents a powerful security token that is already laying in most peoples pocket.

In today’s society more people have a mobile phone than a computer and they carry it with them everywhere. The increased use of public computers in internet cafés and libraries makes the computer unsuitable for containing secure authentication information. By using the mobile phone together with a computer its abilities as a security token can be exploited by users to log in securely and effortlessly to any internet service.

3.2 General Architecture

To be able to authenticate users through a mobile phone, certain main components must be in place. Figure 18 shows the main components and the basic architecture needed for the solutions described later to work.

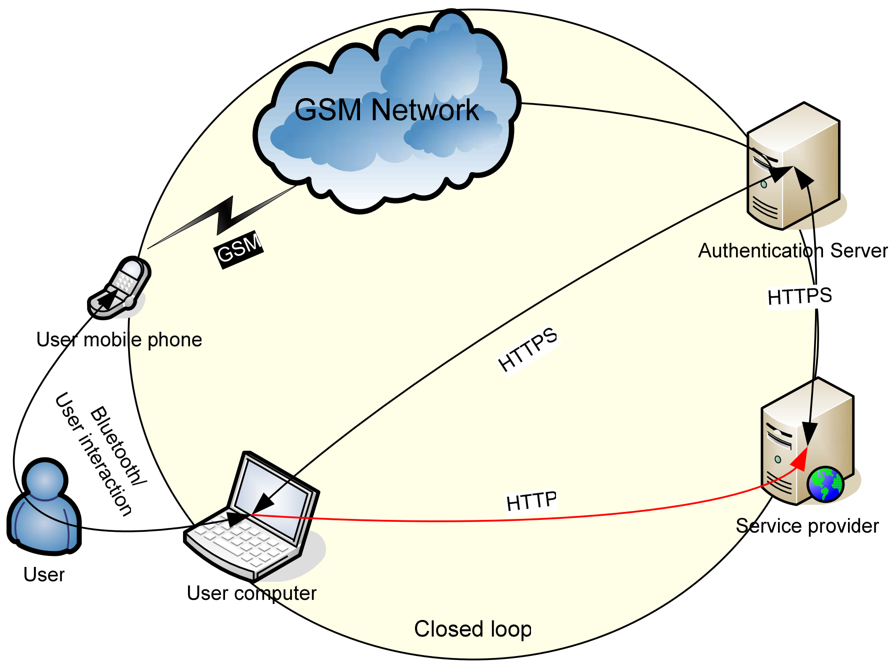


Figure 18. A general architecture of mobile authentication

The user must have access to a computer connected to the Internet and be in possession of a mobile phone with a working SIM card. If the computer and mobile phone is equipped with Bluetooth higher usability can be obtained. Through the Internet browser on the computer the user can access web services provided by service providers. The service provider (SP) is connected to an Authentication Server (AS) that will handle the authentication on behalf of the SP. The AS is connected to the GSM network which enables it to communicate with the user's mobile phone and the operators Authentication Center (AuC). The AS is composed of two parts, an authenticator and an AAA server. The authenticator communicates with the client and relays messages to the AAA server which handles the authentication.

When designing an authentication scheme which uses two separate devices that communicate over two different networks it is very important to ensure that it is the same user that controls both devices. The reason for this is that the authentication path is asymmetric. The request is sent over one communication channel and the response is sent over another. The session on the request channel must be related to the session on the response channel to get a valid authentication. This relation is obtained by ensuring that there is a "closed loop" going through all the components involved in the authentication as illustrated in Figure 18. The loop starts in the device requesting the service, the users computer, goes through the network with SP and AS and then via the mobile phone and back to the initial device either by user interaction or Bluetooth.

This closed loop can be realized in several ways as described in the solutions presented in the next section.

3.3 Authentication solutions

This section presents four possible authentication solutions that use the mobile phone as a secure authentication token implementing the architecture and conditions described above.

3.3.1 SIM strong authentication via mobile phones

This solution makes use of the EAP-SIM protocol to authenticate the user. EAP-SIM is run between the SIM and the AS. The protocol can be run through the computer over Bluetooth and Internet or over the GSM network by SMS. An implementation of this solution has been developed and tested at Telenor [8].

When the user accesses a SP, the browser is redirected to an AS. If the user chooses to run the SIM strong authentication the rest of the procedure is hidden for the user as the SIM card and the AS authenticate each other. If the authentication is successful the browser is redirected back to the SP and the user is logged in.

In order to be able to run the EAP-SIM protocol between the AS and the client the AS needs to communicate with the SIM card. To avoid having to install specific software on the client's computer this communication is handled by a Java applet [33].

The applet will play the role as supplicant and run in the client's browser and relay messages between the authenticator and the SIM card. Its main functions will be to:

1. Receive EAP request from the EAP authenticator

2. Send the contents of the these packets to the SIM card
3. Build EAP response packets from the SIM responses
4. Send the EAP response packets to the EAP authenticator

The applet communicates with the SIM card using Bluetooth SIM Access Profile (SAP). The EAP authenticator is implemented as a Java Servlet running inside the AS. The authenticator first request an EAP identity from the supplicant. The supplicant translates this request and relays it to the SIM card. The SIM card responds with the international mobile subscriber identity (IMSI). The authenticator sends the identity received to the AAA server which contacts the user's AuC for GSM triplets. The challenges contained in the triplets are concatenated and sent back to the supplicant. The supplicant relays the challenges to the SIM card that:

1. Authenticates the server by verifying that the MAC1 received is in order
2. Runs the GSM algorithms to calculate a MAC2 that is sent back to the AS

The supplicant receives the response from the SIM and sends them to the authenticator in EAP format. The authenticator relays the response to the AAA server that verifies that the MAC2 is correct. If MAC2 is correct the authentication is successful and the user is logged in. The EAP-SIM authentication is shown in Figure 19.

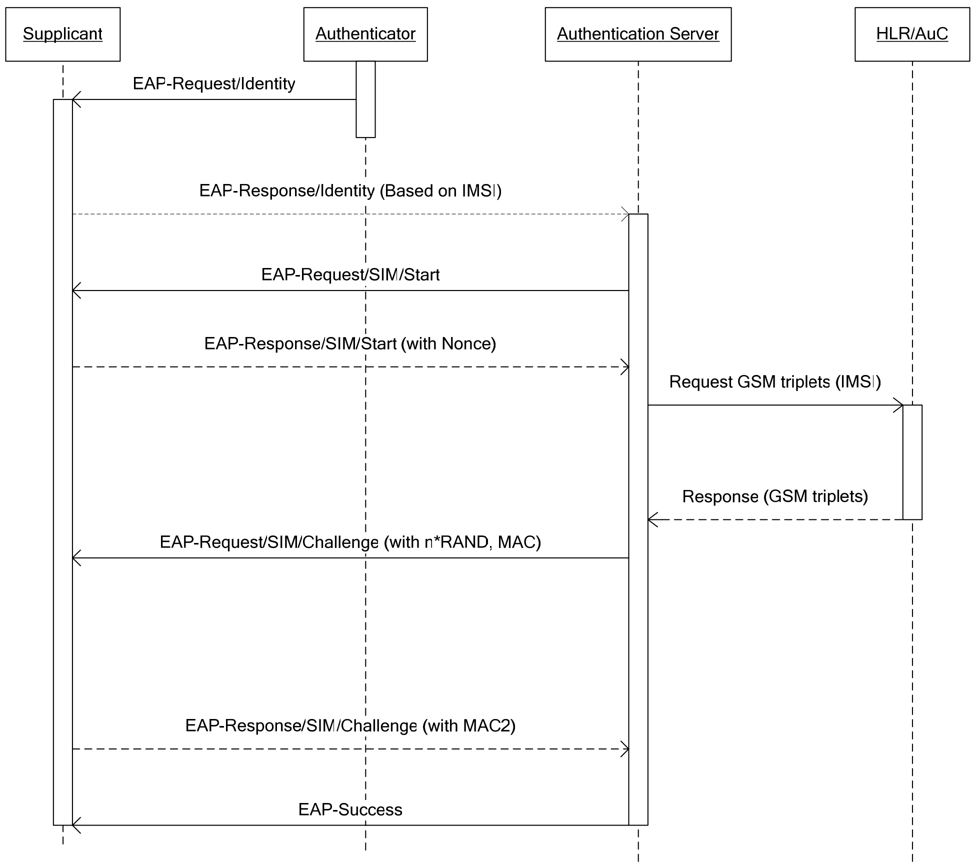

Supplicant
Authenticator
Authentication Server
HLR/AuC
EAP-Request/Identity
EAP-Response/Identity (Based on IMSI)
EAP-Request/SIM/Start
EAP-Response/SIM/Start (with Nonce)
Request GSM triplets (IMSI)
Response (GSM triplets)
EAP-Request/SIM/Challenge (with n*RAND, MAC)
EAP-Response/SIM/Challenge (with MAC2)
EAP-Success

Figure 19. EAP SIM authentication

SMS variant

This solution is a variant that does not require a Bluetooth enabled mobile phone. Instead the EAP-SIM protocol will run over SMS.

The user chooses to log in using the EAP over SMS solution. When this is done a web page is presented asking for a session ID. An authentication applet on the SIM card is started by the AS through SAT and the user is asked to confirm that he wants to start the authentication. A session ID is displayed one the mobile phone screen and the user enters the session ID in the browser. If the session ID is correct the user must accept to start authentication by pressing an OK button on the phone. Then EAP-SIM authentication is performed between the SIM card and the Authentication server by exchanging SMS messages. If the authentication is successful, the SP is notified that the user with the corresponding session ID is authenticated, and the user is provided access to the protected resources.

To enable the phone to perform EAP authentication over SMS an applet on the SIM card must be installed. The applet generates the session ID to be entered in the browser. Thereafter, the SIM applet performs authentication towards GSM using SMS messages which encapsulate the EAP-SIM protocol. SAP is used to access the SIM card and mobile phone during authentication.

The implementation can be optimized so that only two mobile-originated short messages and one server-originated short message are required for full EAP-SIM authentication.

3.3.2 SMS Authentication with Session-ID check

This solution exploits that a user with a valid mobile subscription is already authenticated through the GSM system. Session Ids are used to ensure that it is the same user that controls both the computer and mobile phone.

When the user accesses a service provider a unique session ID is created and sent both to the user’s computer and mobile phone. The session ID is sent over the Internet to the computer and shown in the web browser and sent to the phone by SMS. Then the user can confirm that the session IDs match and send a confirmation by SMS to the service provider. When receiving the confirmation from the user the AS knows that the user is in possession of the phone and the authentication is successful.

The comparison of the session ID can be made by the user or automatically by software if the phone is connected to the computer by Bluetooth.

User verification of session ID

The user accesses the service provider through the Internet browser and chooses to be authenticated by his mobile phone. The user identifies him self with his mobile phone number and the request is redirected to the authentication server. The authentication server then creates a unique session ID and sends this to the user by SMS and over the Internet to the computer. The user checks that the session ID in the SMS is the same as the one shown in the browser. If this is the case the user replies by sending an SMS back to the AS confirming that the session IDs match. When receiving the confirmation the authentication server redirects the browser back to the service provider and the user is given access to the service. The message exchange is shown in Figure 20.

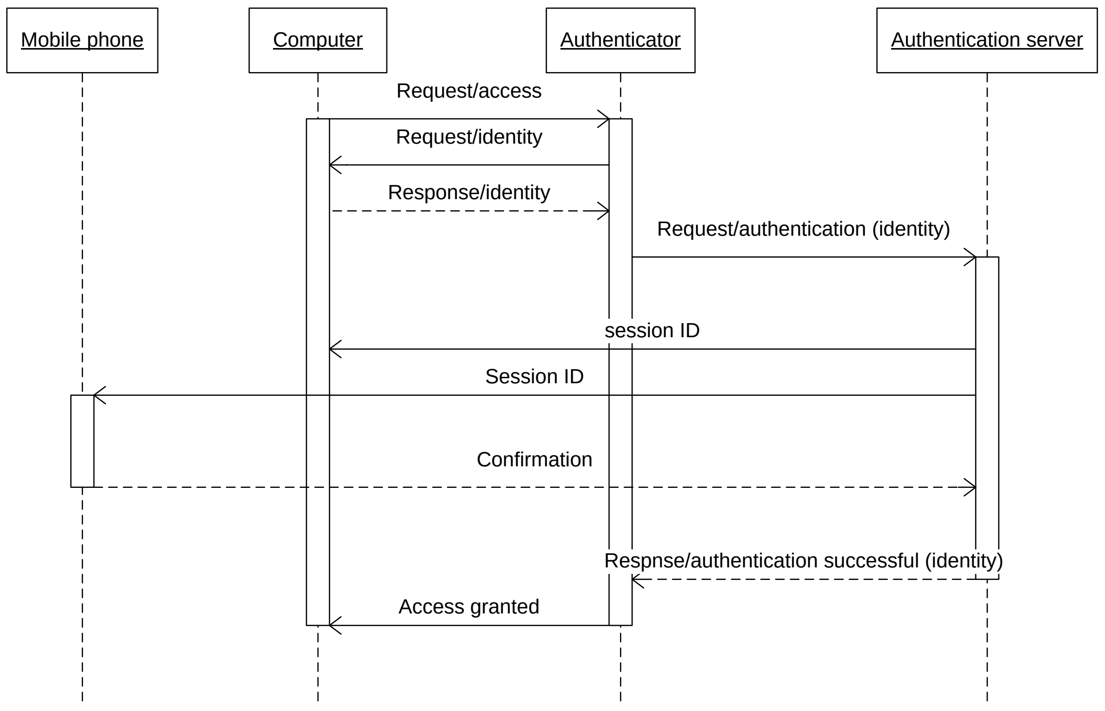


Figure 20. Session ID check

Automatic check of session ID

To make it even easier for the user the session ID can be checked automatically. This can be done by pairing the mobile phone and the computer by Bluetooth.

When the computer receives the session ID from the AS a Java applet running in the browser contacts the SIM card using the Bluetooth SAP. SAP requires that a 16 digit pass-phrase is used during the pairing process. The applet checks the SIM for an SMS with an appropriate session ID. For this to function it must be ensured that the SMS received from the AS is stored on the SIM. This can be done by setting the TP-PID in the SIM header to require SIM data download[34]. This forces the mobile phone to store the SMS on the SIM. If a matching session ID is found the applet creates a confirmation message that is sent to the AS by SMS. This is done with a proactive SIM which can issue commands to the mobile phone as described in[24]. When the AS receives the confirmation it notifies the authenticator that the user is authenticated and redirects the browser back to the SP. The message exchange is show in Figure 21.

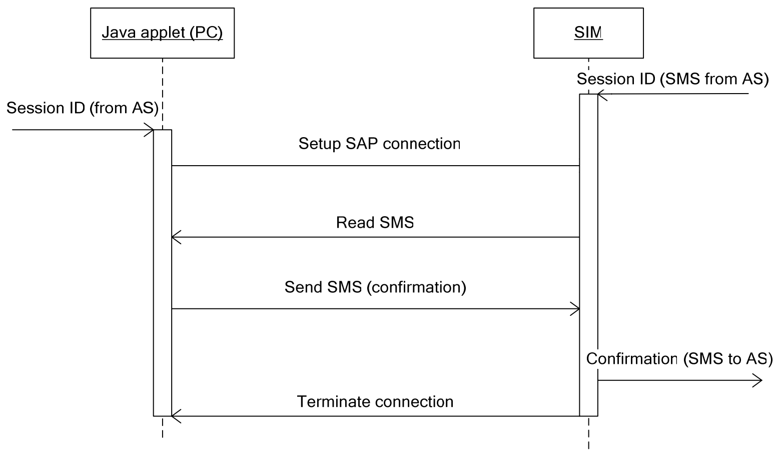


Figure 21. Automatic check of session ID

3.3.3 One-Time-Password from PC to SMS

The next two solutions make use of the OTP principle for authentication. The solutions describe different ways to securely use the mobile phone as an OTP token.

The authentication procedure for the OTP from PC to SMS solution is shown in Figure 22. When the client wants to access the SP the client's identity is requested. The client responds by typing his username in the browser. The message is relayed to the AS which handles the authentication. Upon receiving the client's identity the AS generates a challenge, typically a random number based on the client's profile, and a corresponding OTP. Then the AS sends the challenge to the client. The client enters the challenge on the mobile phone. The OTP applet on the SIM card generates an OTP from the challenge. The OTP is then sent to the AS by SMS. The AS compares the calculated and received OTP and notifies the authenticator that the client is authenticated. The browser is redirected back to the SP and the user is successfully logged in.

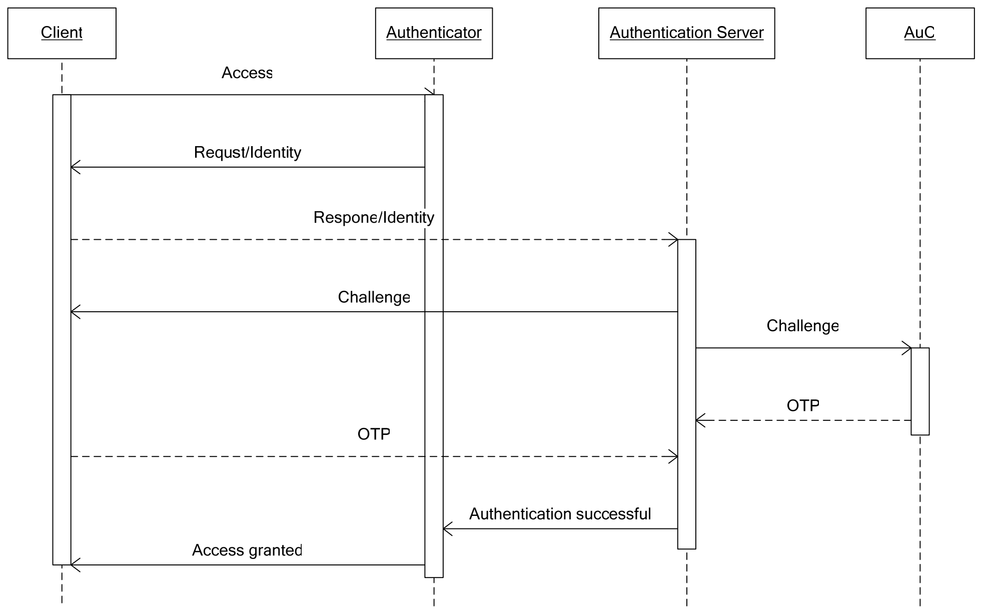


Figure 22. OTP from PC to phone

Manual variant

When the client receives the challenge from the AS it is displayed in the browser. The user starts a MIDlet on the phone which can communicate with the SIM card through the SATSA-APDU as shown in Figure 23. The user is prompted for the challenge and enters it on the phone. The MIDlet transfers the challenge to the OTP applet which responds with an OTP. The user creates an SMS containing the OTP and sends it to the AS.

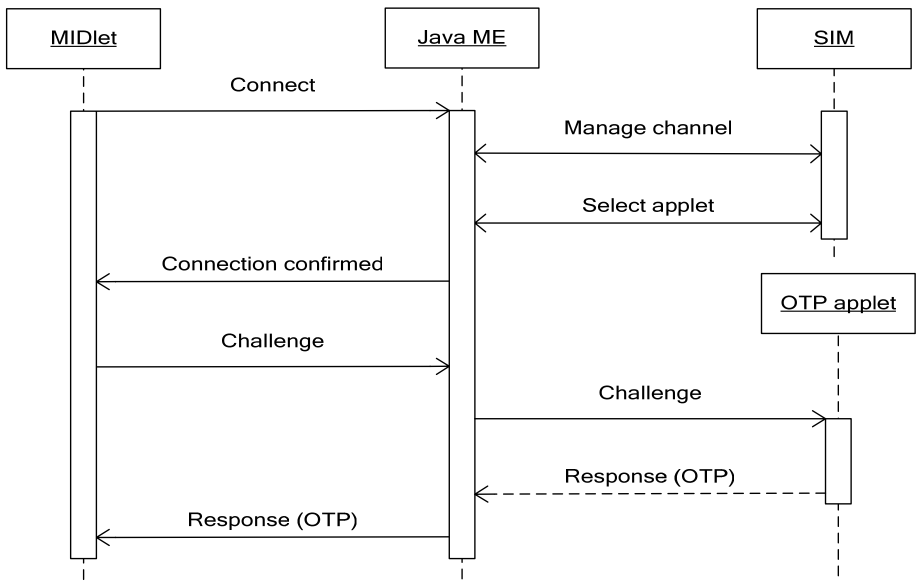


Figure 23. OTP Applet

Automatic variant

If the mobile phone and computer are connected through Bluetooth the challenge can be sent to the phone automatically. To realize this, a Java applet will run in the browser and communicates with the Java MIDlet on the mobile phone. The MIDlet on the phone also has to be expanded so it can create and send an SMS with the OTP to the AS. This can be done by the Wireless Messaging API.

When the user wants to log in using the automatic solution the first thing to be done is pairing the mobile phone and the computer. If this is the first time these two devices are paired a pass-phrase must be entered on both devices. When the pairing is done the user can chose to log in with the automatic solution and after providing an identity the rest of the procedure will go automatically.

After providing the identity a challenge is sent to the user. The Java applet retrieves the challenge and contacts the MIDlet on the mobile phone. When the Bluetooth connection is set up the challenge is sent to the MIDlet. The MIDlet contacts the SIM card as shown in Figure 23. When the OTP is created the MIDlet creates an SMS containing the OTP and sends it through the GSM network to the AS

3.3.4 One-Time-Password from SMS to PC

This solution builds on the same principle as the session ID check, that a user with a working phone is already authenticated through the GSM network. The difference is that the check is done by the server and therefore relieves the user from this burden.

Manual Solution

The user starts the authentication procedure by entering his username. The session is redirected to the AS which creates an OTP based on the users identity by a cryptographic hash function. The OTP is then sent to the user by SMS. When receiving the SMS the user types the OTP in the browser. The AS verifies that the OTP is correct and redirects the browser back to the service provider and the user is logged in. Figure 24 shows the message exchange of this solution.

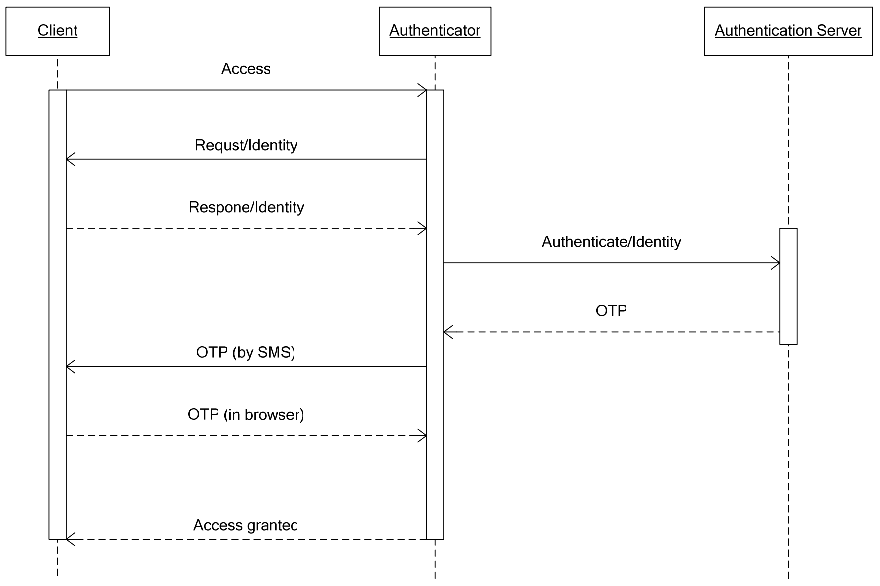


Figure 24. OTP SMS to PC

Automatic variant

If the mobile phone and the computer are paired through Bluetooth the OTP can be transferred automatically from the phone to the computer and then forwarded to the AS through the browser. This can be handled by a Java applet on the computer communicating with the SIM card through SAP. When the AS has sent the OTP by SMS it notifies the client that this has been done. When receiving this notification the Java application contacts the mobile phone and retrieves the SMS from the AS. The applet retrieves the OTP from the SMS and sends it to the AS through the Internet browser. As in the session ID solution the TP-PID field in the SMS header must be set to “SIM DATA DOWNLOAD” so the SMS is guaranteed to be stored on the SIM card. The message exchange is shown in Figure 25.

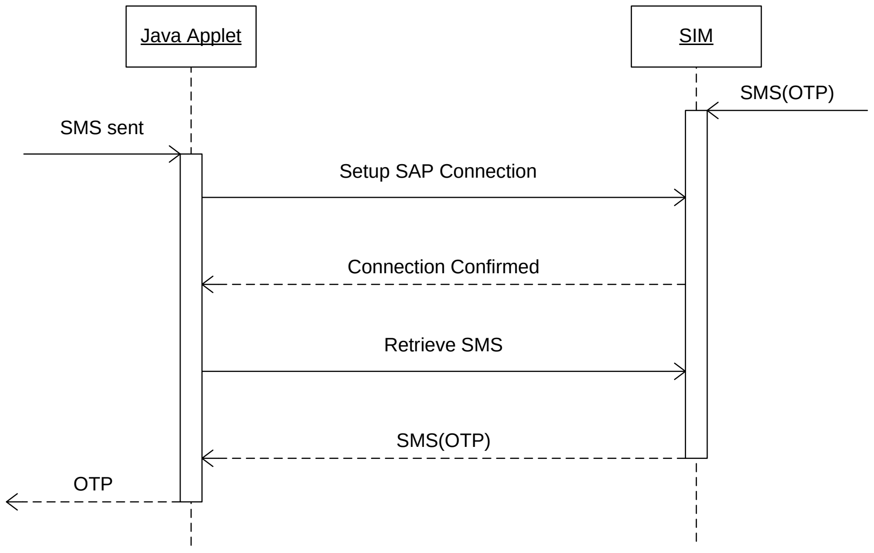


Figure 25. OTP from SMS to PC by Bluetooth

3.4 Selection of a solution

The rest of this thesis will focus on further analyzing and designing one authentication scheme. This work will lead to an implementation of a prototype which can be tested in a real environment. When choosing a solution one important challenge stood out. Is it possible to make a secure solution where the SIM card can be used as it is? If the SIM card does not have to be replaced the deployment cost and time for the solution can be reduced drastically. A new user to the system owning a mobile phone should be able to get authenticated within minutes after the first request, while if the SIM card must be replaced it will take days.

From the four solutions presented above the OTP principle stands out as a promising candidate for such a solution. The simplicity and security properties of OTP make it suitable for implementation on a device with limited resources. The solution that will be further elaborated is therefore a variant of the two solutions presented above. The generation of the OTP is done by the mobile phone, but the SIM card is used as it is.

4 Analysis

This chapter presents the mobile OTP solution. System requirements, use cases and interaction diagrams are presented here to help clarify the required functionality, components and interfaces.

4.1 Requirements

Requirements are an important factor when designing software solutions. They describe what the system should do and how it should behave. Requirements must be measurable, testable and defined to a level of detail sufficient for system design. Requirements are divided into functional requirements and non functional requirements.

4.1.1 Security functional requirements

Functional requirements capture the intended behavior of the system [35]. In an authentication system the security functionality is the main objective. Below the security functional requirement of the system are shown.

– Passwords must be secured against online guessing, such as dictionary attacks.
– Messages and authentication assertions must be protected against replay attacks
– The authentication protocol must prevent eavesdropping
– Shared secrets must not be revealed to third party
– A minimum of two-factor authentication must be used
– Mutual authentication must be supported
– The protocol should be protected against session hijacking
– All sensitive data should be encrypted

4.1.2 Non functional requirements

Non-functional requirements describe constraints and qualities of a system [36]. The non-functional requirements are shown below.

Usability: The solution must offer the same or better usability than existing systems. They must not require special knowledge from the users, or demand expensive software or hardware on the user side. The solutions should work together with single sign-on systems.

Deployment cost: The solutions should not be more expensive to deploy than existing solutions. They should not require any extra expenses for the users.

Availability and reliability: The authentication system must be available whenever the user needs to be authenticated. The error rate must not be higher than for existing systems.

Scalability: The system must scale easily when the number of users increases.

Performance: The authentication delay must be in the same range as that of existing solutions.

4.2 Use Case Diagrams

Use cases are a technique for capturing the functional requirements of a system[37]. This chapter presents several use cases to help identify the requirements of the system. Both high level and lower level use cases are used.

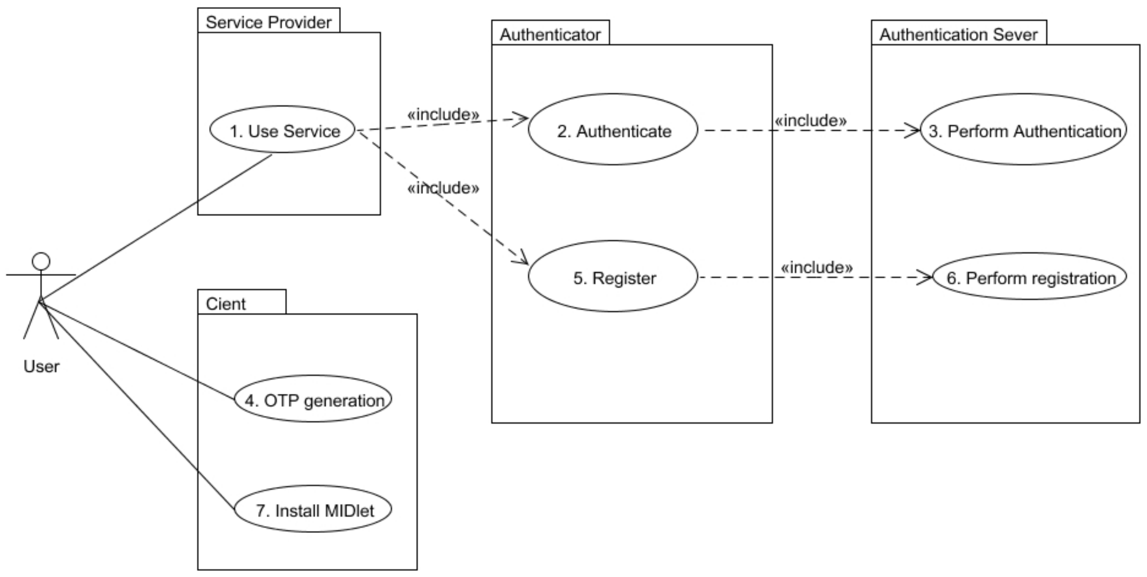


Figure 26. Use Case – Overview

Figure 26 gives an overview of the main use cases. In Table 2 to Table 4 textual descriptions of the main scenarios “authentication” and “registration” are given. Both scenarios start with a user accessing a service which requires authentication. When the user interacts with the system the user takes the role as actor, even though the communication always goes through the client. When the client components communicate without user interaction the client is the actor.

Table 2. Use case - Use Service

Use Case	1. Use Service
Description	A user requests access to a service that requires strong end-to-end authentication.
Actors	User Service Provider (SP) Authenticator
Assumptions	The SP needs to be connected to an Authenticator The client needs to provide the correct credentials
Steps	1. The user tries to access the service 2. The SP redirects the request to the Authenticator 3. The Authenticator starts the authentication process

	(Start UC2: “Authenticate”) 4. The Authenticator vouches for the user 5. The SP gives the user access to the service
Variations	3a. The client is not registered (Start UC4: “Register”) 4a. The Authenticator refuses to vouch for the user and access to the service is denied
Issues	

Table 3. Use case - Authenticate

Use Case	2. Authenticate
Description	A client is authenticated by the mobile OTP solution
Actors	User Client Authenticator
Assumptions	The SP needs to be connected to an Authenticator The client needs to provide the correct credentials
Steps	1. The User is asked for a username 2. UC3 "perform authentication" is started 3. The Authenticator sends a challenge and MAC to the Client 4. The Client calculates the OTP and sends is back 5. The Authenticator validates the OTP received form the Client
Variations	4a. The MAC is incorrect and authentication fails 5a. The OTP is invalid and the authentication fails
Issues	

Table 4. Use case - Register

Use Case	5. Registration
Description	A user is registered to the mobile OTP service
Actors	User Authenticator
Assumptions	The SP needs to be connected to an Authenticator The user needs to provide valid user data
Steps	1. An unregistered user tries to access the service 2. The User is asked to enter personal information 3. UC5 "perform registration" is started 4. The User is notified that the registration is completed 5. The User can download the OTP MIDlet 6. UC7 is started

Variations	4a. The registration fails
Issues	

To further clarify the functionality of the system use case 3, 4, 6, and 7 from Figure 26 are described in more detail.

4.2.1 Use Case 3 – Perform authentication

The use case "perform authentication" describes what happens at the Authentication Server (AS) when a client is authenticated. As can be seen in Figure 27 use case 3 includes three other use cases that further describe the authentication process.

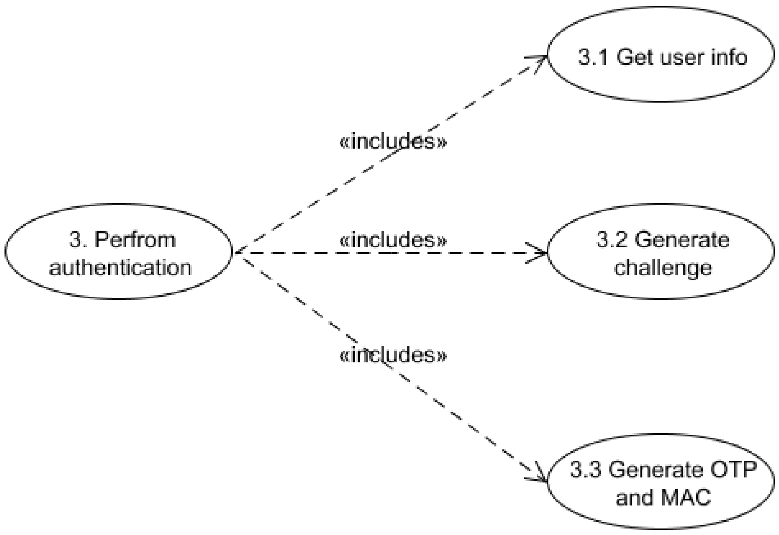


Figure 27. Use case- server side

Table 5. Use case - Perform authentication

Use Case	3. Perform authentication
Description	The Authenticator contacts the AS on behalf of a user which needs to be authenticated
Actors	AS Authenticator
Assumptions	The user has provided a valid username

Steps	1. The AS gets user data for database 2. The AS generates a challenge, OTP and MAC 3. The authentication triplet (challenge, OTP, MAC) is returned to the Authenticator
Variations	1.a The username is invalid and the AS can not authenticate the user
Issues	

Table 6. Use case - Get user info

Use Case	3.1 Get user info
Description	The Authentication Server gets user info from user data base
Actors	Authentication Server (AS)
Assumptions	The client has provided a valid username
Steps	1. The AS retrieves the user profile for the given username and returns the secret key and phone number.
Variations	1.a An invalid username is given and the user profile can not be retrieved.
Issues	

Table 7. Use case -Generate challenge

Use Case	3.2 Generate challenge
Description	The AS generates a challenge
Actors	AS
Assumptions	The client has provided a valid username
Steps	1. The AS generates a challenge is created based on the user profile
Variations	
Issues	

Table 8. Use case - Generate OTP

Use Case	3.3 Generate OTP

Description	The AS generates an OTP and a MAC
Actors	AS Authenticator
Assumptions	The AS has a shared secret with the client A challenge is already created
Steps	1. The AS generates a OTP from a challenge and secret key 2. Based on the secret key the AS calculates a MAC over this OTP 4. The AS sends the MAC, OTP and challenge to the Authenticator
Variations	
Issues	

4.2.2 Use case 4 – OTP Generation

When the client receives a challenge and MAC from the Authenticator it crates an OTP to send back. Four use cases describe this process as shown in Figure 28.

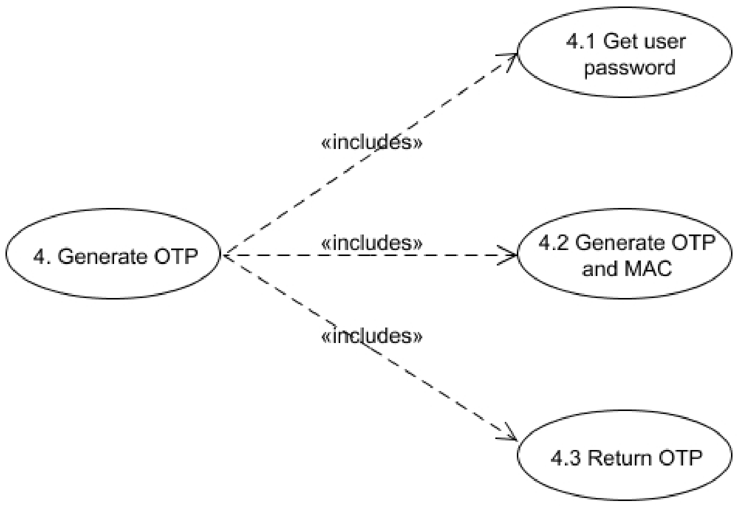


Figure 28. Use case - OTP Generation

In Table 9 to Table 12 the use cases are described in detail.

Table 9. Use case – OTP Generation

Use Case	4. Generate OTP

Description	The client generates an OTP to send back to the Authenticator
Actors	Client Authenticator User
Assumptions	The client has received a challenge from the Authenticator
Steps	1. The User is asked to enter his password (UC 4.1) 2. The Client generates the OTP (UC 4.2) 3. The Client calculates the MAC (UC 4.3) 4. The Client sends the OTP back to the Authenticator (UC 4.4)
Variations	1a. The password is wrong and the OTP generation fails 4a. The MAC is not correct and the OTP is not returned.
Issues	Limit the number of possible tries to type the password?

Table 10. Use case - Get user password

Use Case	4.1 Get user password
Description	The user must enter a password
Actors	User Client
Assumptions	The client has received a challenge from the Authenticator
Steps	1. The Client asks the user for his password 2. The Client checks if password is correct 3. IF password correct THEN decrypt secret key start UC 4.2 Generate OTP ELSE Ask user for his password
Variations	
Issues	Limit the number of possible tries to type the password?

Table 11. Use case – Generate OTP and MAC

Use Case	4.2 Generate OTP and MAC
Description	The client generates an OTP
Actors	Client
Assumptions	The client has received a challenge from the Authenticator The secret key has been decrypted
Steps	1. The Client uses the challenge received from the Authenticator and the decrypted secret key to generate an OTP. 2. The Client uses the secret key to calculate a MAC over the OTP 3. The Client compares the calculated MAC with the MAC received from the Authenticator
Variations	3a. The MAC is not correct. An error message is sent back to the Authenticator instead of the OTP
Issues	

Table 12. Use case - Return OTP

Use Case	4.3 Return OTP
Description	The Client returns the OTP to the Authenticator
Actors	Client Authenticator
Assumptions	The MAC is correct
Steps	1. The Client establishes a secure connection to the Authenticator 2. The Client sends the OTP to the Authenticator
Variations	1.a A secure connection could not be established and the OTP is not returned
Issues	

4.2.3 Use case 6 – Perform registration

A new user to the Mobile OTP system must register. During the registration process a username is chosen and the user is allowed to download an OTP MIDlet that must be installed on the mobile phone (UC7). The use cases for the registration process are illustrated in Figure 29. Table 13 to Table 17 describe the use cases in further detail.

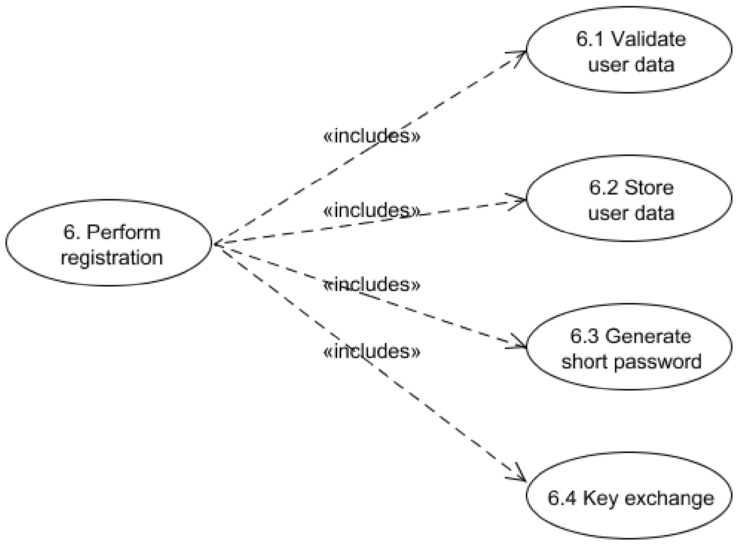


Figure 29. Use case – Registration

Table 13. Use case - Perform registration

Use Case	6 Perform Registration
Description	The AS registers a new user
Actors	AS User Authenticator
Assumptions	A new user has submitted a registration form
Steps	1. The AS checks that the user data is valid (UC 5.1) 2. The AS stores the data in the user database (UC 5.2) 3. The AS generates a short password (UC 5.3) 4. The AS returns the key exchange values and the password to the Authenticator which initiates a key exchange with the Client (UC 5.4)
Variations	1.a The user data is not valid and the registration fails
Issues	

Table 14. Use case - Validate user data

Use Case	6.1 Validate user data
Description	The AS checks that user data is valid
Actors	AS
Assumptions	The user has provided personal data
Steps	1. The AS checks that the User has entered a valid telephone number
Variations	1.a The phone number is not valid and the validation fails
Issues	

Table 15. Use case - Store user data

Use Case	6.2 Store user data
Description	The AS stores the data from the new user
Actors	AS
Assumptions	The user has presented valid user data
Steps	1. The user data is stored in a user database
Variations	
Issues	

Table 16. Use case - Generate password

Use Case	6.3 Generate password
Description	The AS generates a password
Actors	AS Authenticator User
Assumptions	The user has provided valid user data
Steps	1. The AS generates a short password to secure the key exchange 2. The AS sends the password the Authenticator which relays it to the

	User
Variations	
Issues	The link must be secured with TLS.

Table 17. Use case - Key exchange server

Use Case	6.4 Key exchange server
Description	A key exchange is performed between the Client and Authenticator to establish a shared secret key
Actors	Client Authenticator AS
Assumptions	The user has provided valid user data A password has been generated and sent to the User
Steps	1. The AS generates a large prime p 2. Based on p and the password the AS calculates a key pair 3. The AS sends p and the public key to the Authenticator which initiates a key exchange with the Client 4. The AS gets the client public key from the Authenticator and generates the shared secret key 5. The AS stores the secret key in the user profile
Variations	
Issues	Key validation might necessary

4.2.4 Use case 7 – Install MIDlet

The MIDlet must be installed on the user’s mobile phone before the phone can be used as an OTP token. The use cases for the installation of the MIDlet are shown in Figure 30. Some user-interaction is required in the setup of the MIDlet. This only has to be done once when the MIDlet is installed and the user interaction during the authentication is kept at a minimum.

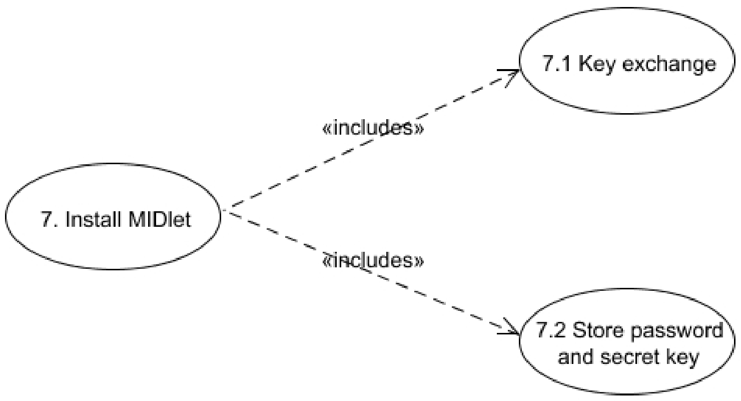


Figure 30. Use case - Install MIDlet

Table 18. Use case - Install MIDlet

Use Case	7. Install MIDlet
Description	The OTP MIDlet is installed on the user's phone
Actors	User Client Authenticator
Assumptions	The user is successfully registered
Steps	1. The Client downloads and installs the MIDlet 2. The MIDlet is automatically started upon receiving a SMS from the Authenticator 2. The MIDlet prompts the user for two passwords. 3. The MIDlet performs the key exchange with the Authenticator 4. The MIDlet notifies the user that the key exchange is complete
Variations	
Issues	

Table 19. Use case - Store password

Use Case	7.1 Key exchange
Description	key exchange is performed between the client and the Authenticator
Actors	Client

	Authenticator AS
Assumptions	The MIDlet is downloaded to the phone The correct password is entered on the phone
Steps	1. The Client retrieve the key values from the SMS received from the Authenticator 2. The client generates its own public values and sends them to the Authenticator. 3. Based on the public values the client and AS calculates the same secret key.
Variations	
Issues	It might be necessary to include key validation

Table 20. Use case - Key Exchange

Use Case	7.2 Store password and secret key
Description	The password and secret key are stored
Actors	Client
Assumptions	The MIDlet is downloaded to the phone The correct password is entered on the phone
Steps	1. The Client encrypt the secret key with the user password as encryption key 2. The Client generates a hash of the user password 3. The Client stores the encrypted secret key and the password hash in the mobile phone record store.
Variations	
Issues	

4.3 Interaction diagrams

Interaction diagrams are used in UML to describe how objects collaborate together[37]. The most common and intuitive interaction diagram is the sequence diagram. Sequence diagrams capture the behavior of a given scenario and are very useful for describing the message exchange in a system. Three sequence diagrams are presented showing the registration, key exchange and authentication scenario.

4.3.1 Registration

Figure 31 shows the registration of a new user. The registration is started when an unregistered user wants to use the mobile OTP solution for authentication. The user is asked to type in personal data and phone number. The data is relayed to the AS which checks that the phone number is valid and then stores the data in the user data base. A short password is created and exchanged with the user through the Internet browser. The user can now download and install the OTP MIDlet. When this is done the AS initializes an asymmetric key exchange to establish a strong shared key with the client.

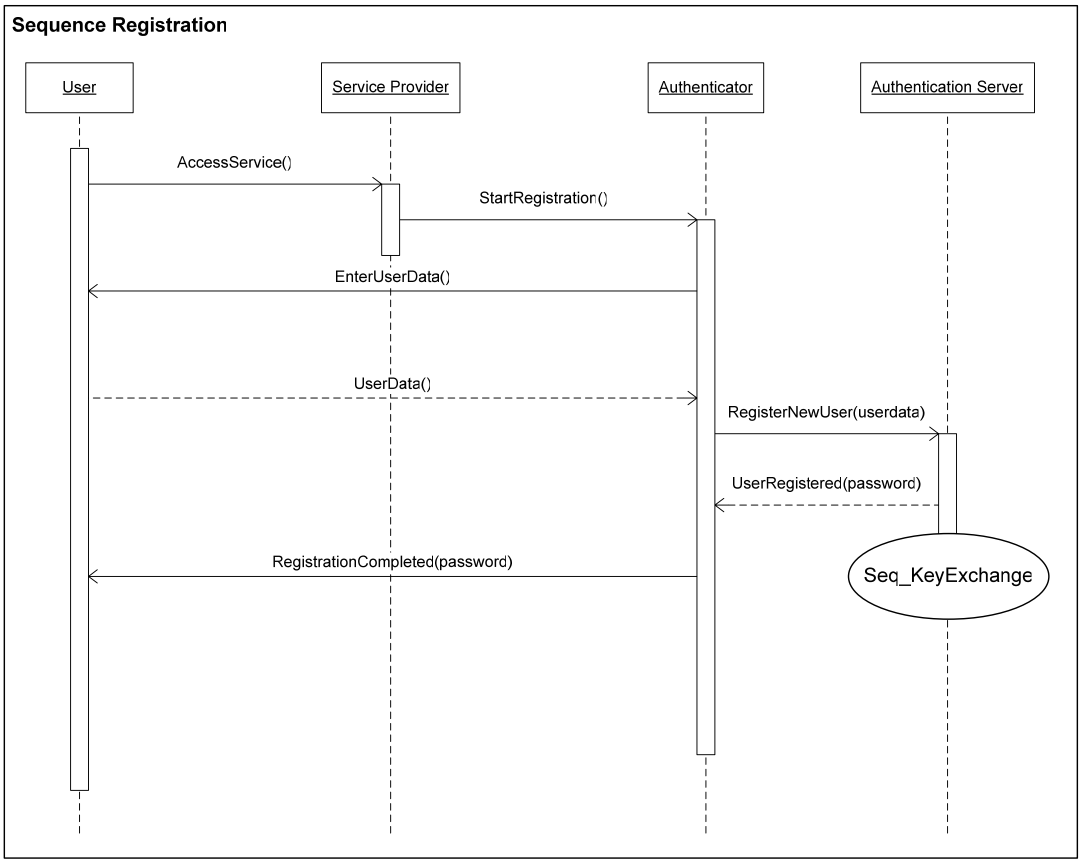


Figure 31. Sequence Diagram – Registration

4.3.2 Key Exchange

The Simple Password Exponential Key Exchange (SPEKE) protocol [38] is used for the key exchange. SPEKE is an authenticated version of Diffie-Hellman (DH) [39] which prevents Man-In-The-Middle attacks by basing the DH generator on a hash of a shared password. An attacker who is able to read and modify all messages between the client and server cannot learn the shared key and cannot make more than one guess for the password in each interaction with a party that knows it.

The key exchange is shown in Figure 32 and as can be seen only two messages is necessary to complete the exchange. The parameters used in the key exchange are defined below

p– a large and randomly selected safe prime
g – the Diffie-Hellman generator
π – short password shared between the client and server
a – server private key
k_a - server public key
b – client private key
kb- client public key
k – shared secret key

The AS starts the key exchange by generating p. Based on p and a hash of π, g is calculated

$$g = \text{hash}(\pi)^2 \bmod p$$

Then the AS computes a, which is a secret random integer. Now k_a can be calculated

$$k_a = g^a \bmod p$$

When this is done AS sends p and k_a to the Authenticator. The Authenticator sends the values on to the client with an SMS. Upon receiving the SMS from the Authenticator the MIDlet is automatically launched and the user is asked to enter two passwords. The first password is a user chosen password used to ensure two-factor authentication. The second password is π and is used to authenticate the key exchange. π is displayed in the browser when the registration is completed.

When the user has entered the passwords the MIDlet retrieves p and k_a from the SMS and calculates the same generator g as the AS.

$g = \text{hash}(\pi)^2 \bmod p$

Then the MIDlet generates a secret random integer to be used as band calculates the public key k_b.

$k_b = g^b \bmod p$

When k_b is calculated the MIDlet sends it back to the Authenticator and computes the shared secret key

$k = (k_s)^b \bmod p.$

When AS receives k_b from the Authenticator it computes

$k = (k_b)^a \bmod p$

Now the AS and the MIDlet shares a strong secret key k which can be used for generating the OTP.

The MIDlet completes the installation by encrypting the shared key with the user's personal password and then storing both the encrypted key and a hash of the password in the phone's memory.

When installation is complete the user can be authenticated by the mobile OTP solution.

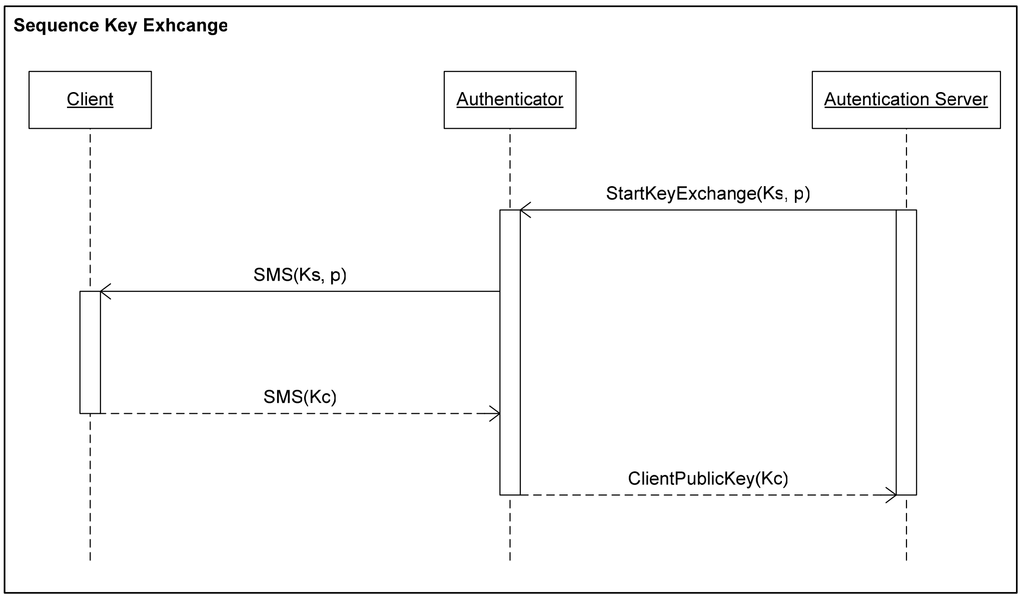


Figure 32. Sequence Diagram - Key Exchange

4.3.3 Authentication

Figure 33 shows a successful authentication of a client using the mobile OTP solution. It is required that the user is already a registered user of the authentication service and that the OTP MIDlet is installed on the user's mobile phone. The authentication is initiated when a user requests access to a service that requires authentication. The SP notifies the authenticator that a user needs to be authenticated. The session is redirected to the authenticator and the user is asked to enter a username. The username is sent to the AS which gets the secret key for this client and from this generates an OTP. The OTP is also based on a challenge. A different challenge is used every time so the generated OTP is always changing. At last a message authentication code (MAC) based on the secret key is calculated over the OTP. The AS sends the triplet (challenge, MAC, OTP) to the Authenticator which relays the challenge and the MAC to the client. Upon receiving the challenge the client calculates the OTP. Then it calculates the MAC and compares it to the one received from the Authenticator. If the values match the client can authenticate the AS since the AS has proved that it is in possession of the shared key. The client then sends the OTP back to the Authenticator. If the MAC is wrong the authentication is aborted. The Authenticator compares the OTP with the one received from the AS and if they match notifies the SP that the client is authenticated. A mutual authentication of the client and server has been achieved and the session is redirected back to the SP which grants the user access to the service.

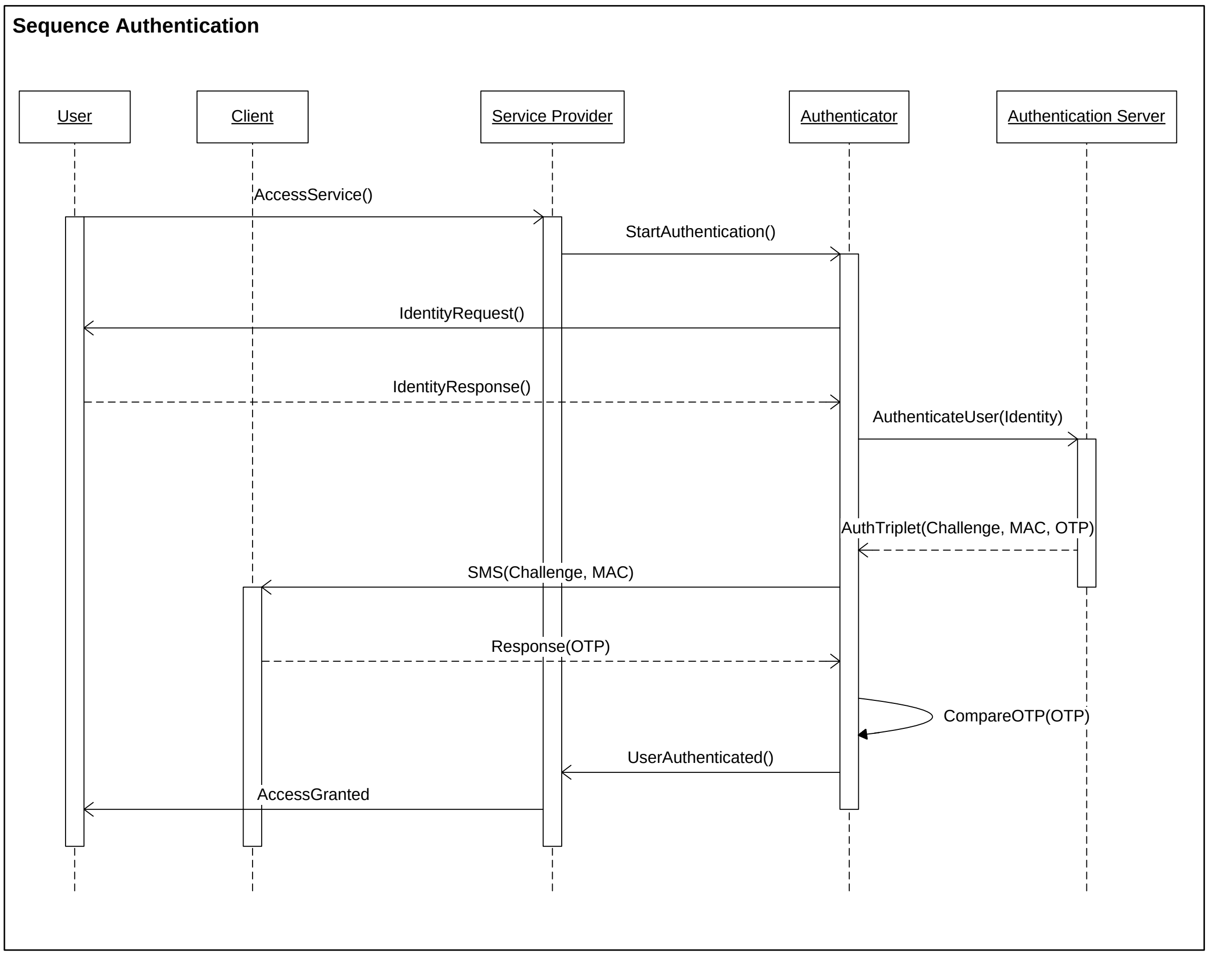


Figure 33. Sequence Diagram - Full authentication

4.4 Summary

This chapter has presented a thorough analysis of the mobile OTP system. Requirements, use cases and interaction diagrams are used to identify the functionality, components and interfaces required to realize the Mobile OTP authentication system.

5 Design

The design process is important to identify which components, packages and classes the system will consist of.

5.1 Components

Components represent individual pieces of the system that are independent from each other [37].

The main components of the system are illustrated in Figure 34. As can be seen the system consist of four main components. The client side consists of the OTP module and a server proxy. The server side is divided in two parts. The Authenticator handles the communication with the client while the AS handles the authentication. Each one of the main components will be described in detail in this chapter.

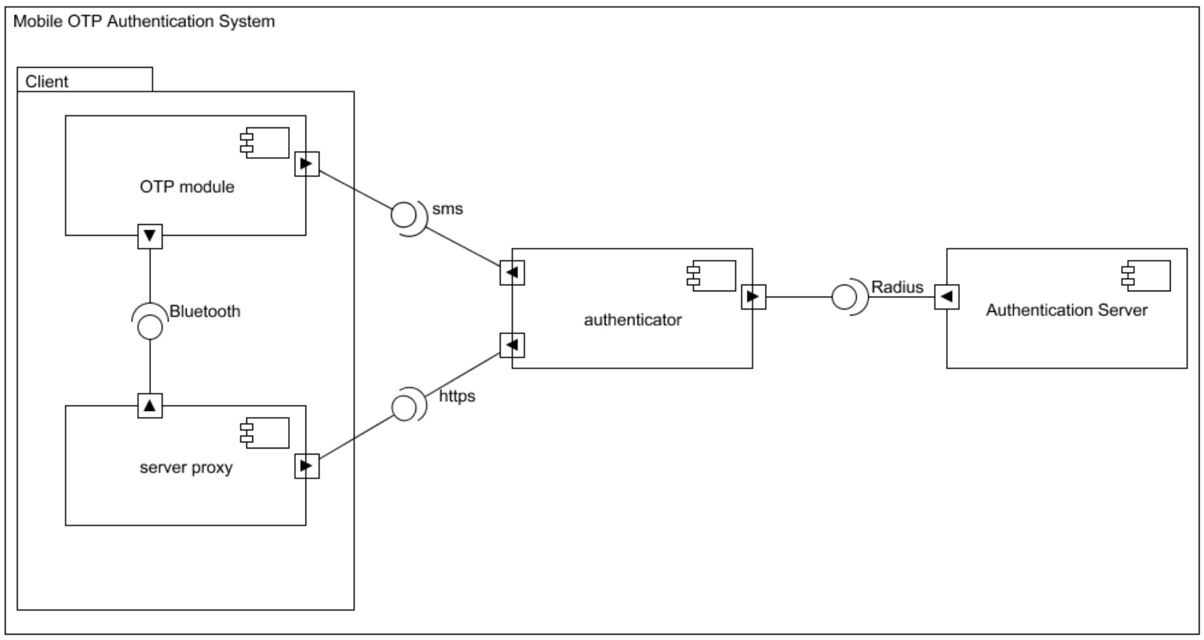


Figure 34. Main component diagram

The OTP module communicates with the Authenticator with short messages over the GSM network. The module also has a Bluetooth interface for communicating with the server proxy running on client computer. The server proxy is connected to the Authenticator over HTTPS. The Authenticator communicates with the Authentication Server over RADIUS.

5.1.1 OTP Module

The OTP module runs on the mobile phone and handle the creation of the OTP when requested. The module has two communication interfaces, SMS communication towards the Authenticator and Bluetooth communication towards the Server Proxy. The OTP Module consists of the Core component and the Crypto component.

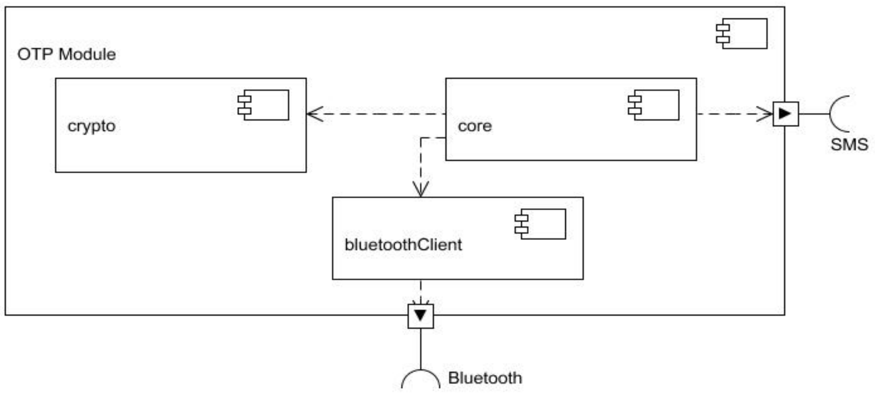


Figure 35. Component -OTP Module

5.1.2 Server proxy

The server proxy runs in the browser on the user's computer and act as a Bluetooth server. Its main objective is to relay the OTP from the client to the Authenticator. The Server Proxy is initiated when the authentication is started. It opens a Bluetooth connection which the OTP module can connect to. When the OTP from the module is received an HTTPS connection towards the Authenticator is opened and the OTP is relayed on. The Server Proxy only consists of one package since it has a very limited operation area.

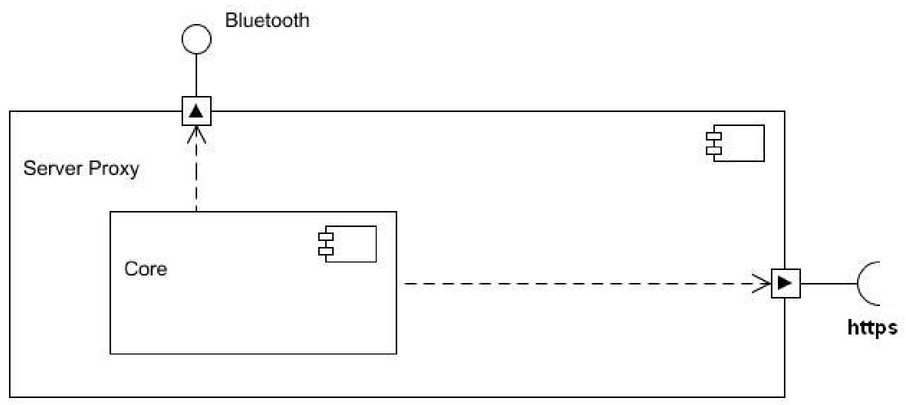


Figure 36. Component - Server Proxy

5.1.3 Authenticator

The Authenticator handles communication with the client and service provider. The Authenticator consists of three main components. The ComInterface handles communication with the client and Service Provider over HTTPS and SMS. The Core is the central component responsible for handling authentication and register request and keeping track of authenticated clients. The Radius Client is required to communicate with the Authentication Server using Radius.

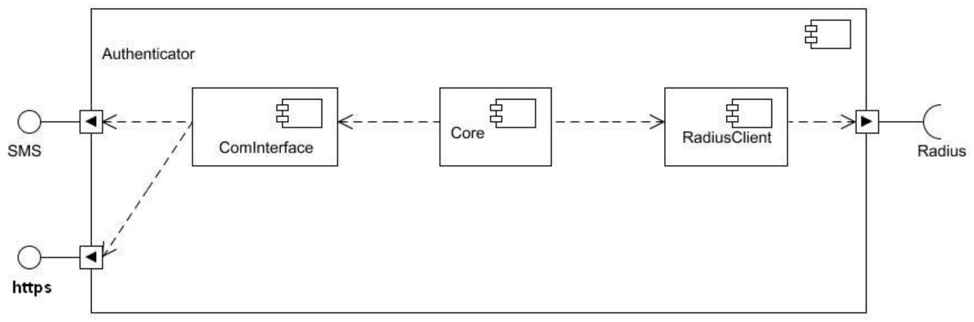


Figure 37. Component - Authenticator

5.1.4 Authentication Server

The Authentication Server handles the registration and authentication. The OTP component is included for handling the creation of the specific OTP parameters and key exchange. The user database can be any commercial database. There already exits many commercial Authentication Servers that can be used by only adding the OTP plugin. The Authentication Server will therefore not be further elaborated.

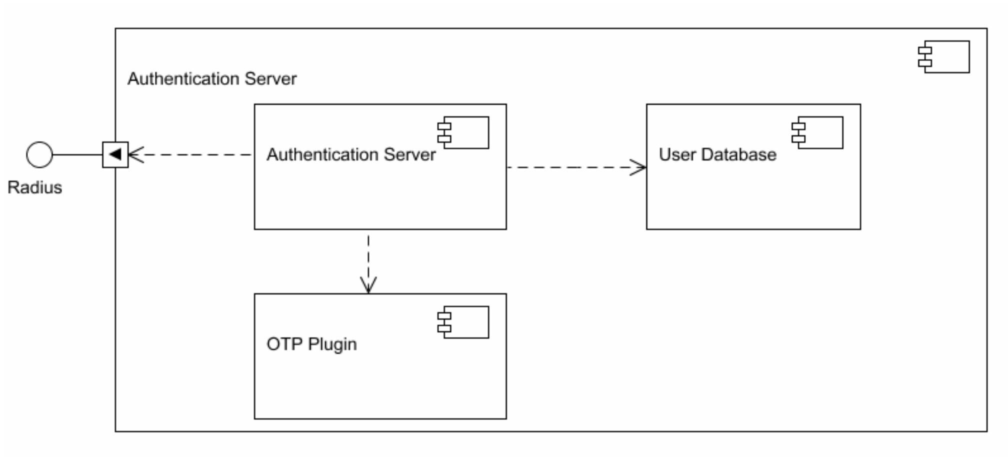


Figure 38. Component - Authentication Server

5.2 Class Diagrams

A class diagram describes the types of objects in a system and the relationships among them. They show the properties and operations of a class and the constraints that apply to the connections between them[37].

Package diagrams are used to group elements together into higher-level units[37]. Classes from the same part of the system are placed in the same group. Figure 39 shows the packages of the authentication system. Class diagrams for the packages OTP module, Server Proxy and Authenticator will be presented below.

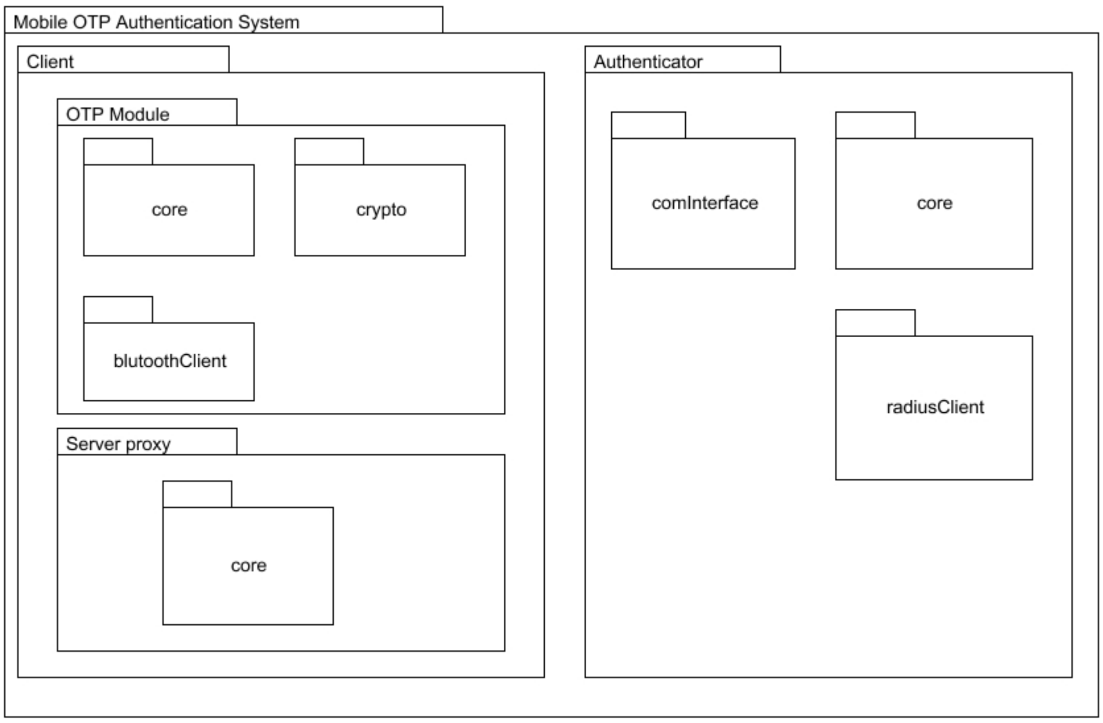


Figure 39. Package Diagram

Figure 40 show the class diagram for the OTP module. The OTP MIDlet class implements the MIDlet functionality and handles the SMS communication with the Authenticator. The MIDlet uses the Bluetooth class to set up a Bluetooth connection with the user's computer and transfer the OTP. The Functions class contains the methods for storing data and generating the OTP. The KeyExchange class contains the methods for performing the key exchange with the Authenticator. The three classes in the crypto package are static and implement the cryptographic algorithms needed.

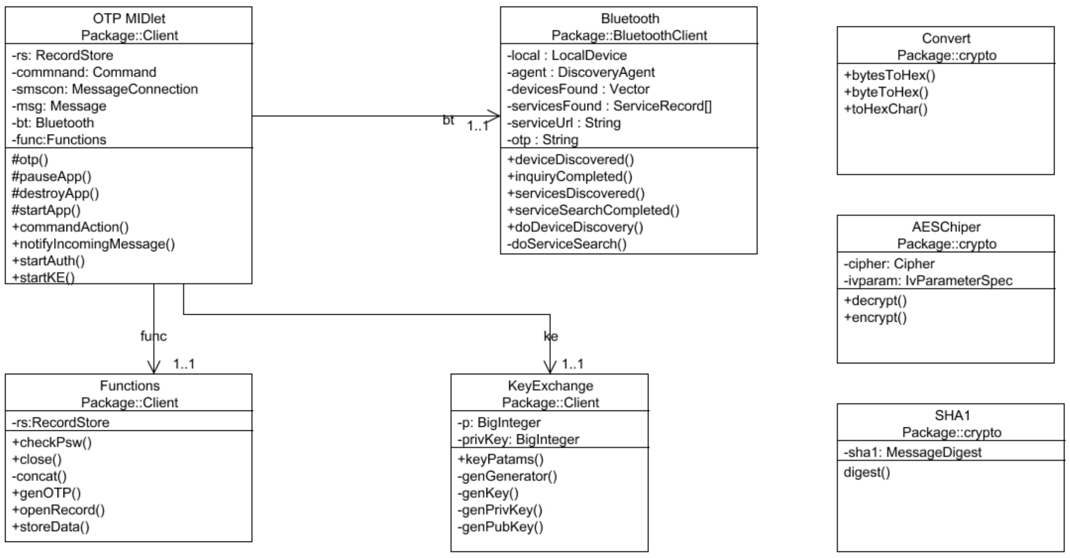


Figure 40. Class Diagram - OTP Module

In Figure 41 the class diagram for the server proxy is shown. The Applet uses the BluetoothServer to setup a Bluetooth server on the computer. The server is discoverable and will accept incoming connection request from a client. When a client has connected it sends the OTP to the server proxy. When the OTP is received the SocketCommunication class sets up a secure socket connection to the Authenticator. The OTP is then transferred to the Authenticator via this connection.

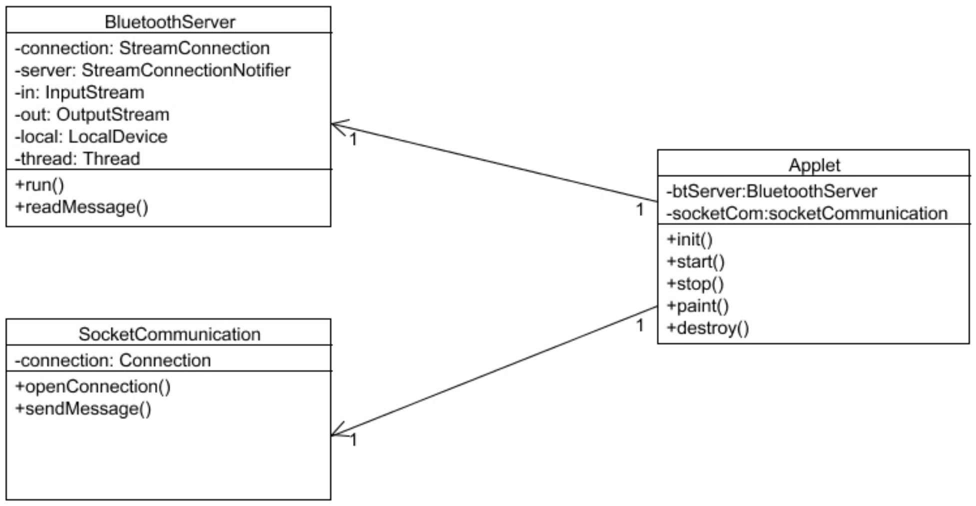


Figure 41. Class Diagram - Server proxy

The classes for the Authenticator package are shown in Figure 42. The AuthenticatorCore class is the central point of the Authenticator and has the manager role. The Communicator handles communication with the client and Service Provider. It provides an interface to the

SMS gateway and handles http request and responses. The RadiusClient handles communication with the Authentication Server translating messages to and from the Radius format. Ongoing client sessions are contained in the ClientSession class.

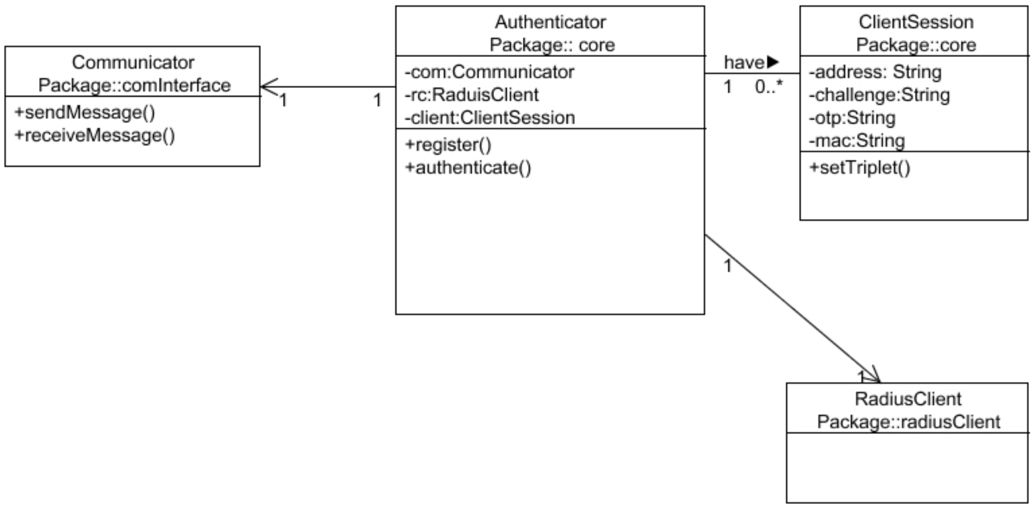


Figure 42. Class Diagram - Authenticator

5.3 Summary

During the work with this chapter the system design has become clearer. Components, packages and classes have been identified. The necessary communication links and interfaces between the different parts of the system are revealed. As shown the system can be developed with existing technologies and protocols and some parts can be realized with existing components. The OTP module, Server Proxy and authenticator must be developed and some parts must be made from scratch.

6 Implementation

This chapter describes the implementation of a prototype showing the main concepts of the solution. First a possible deployment of the system is presented. Then the developed components are described.

6.1 Deployment

Deployment diagrams show the physical layout of a system, revealing which pieces of software run on what pieces of hardware. A possible deployment for the mobile OTP system is shown in Figure 43.

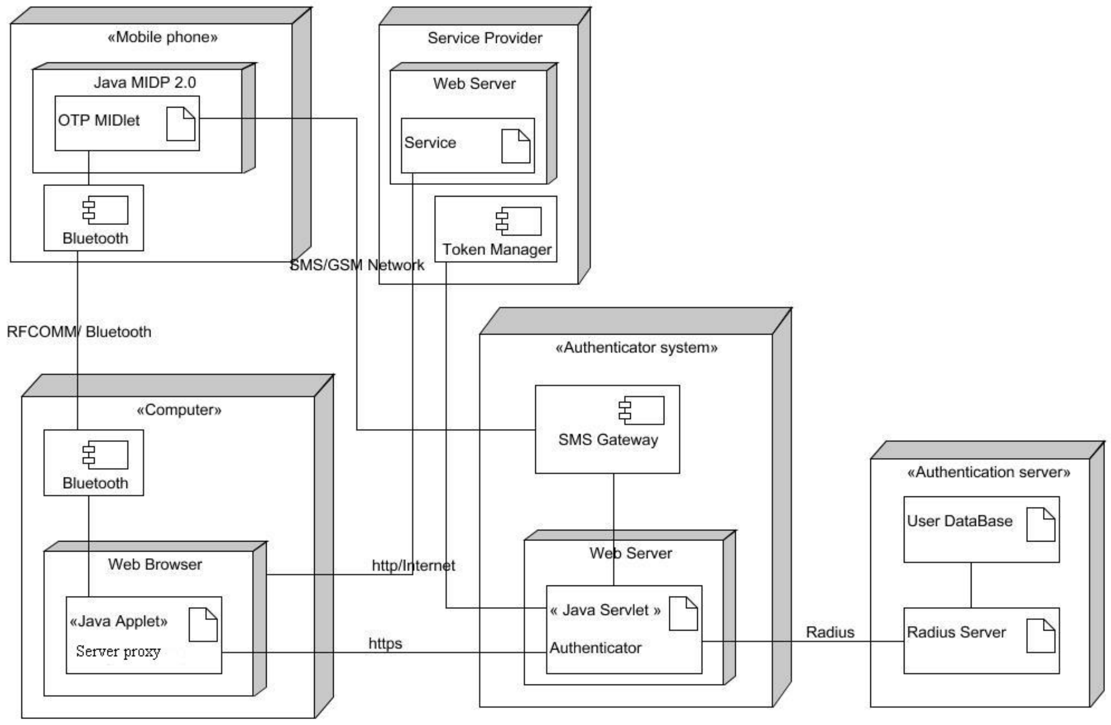


Figure 43. System Deployment

6.2 OTP Module

The OTP Module is a program package that is installed on the user's mobile phone. It is realized as a MIDlet suite that runs in the Java environment on the mobile phone. It is the core component on the client side and enables the mobile phone to act as an OTP token and automatically exchange authentication messages with the AS and computer. The MIDlet will launch automatically upon receiving a SMS message on a specific port. The content of this SMS is retrieved and used for generating authentication values.

6.2.1 MIDlet

As described in chapter 3.4 Java ME is composed of different profiles and configurations. The choice of profile and configuration depends on the device the application is written for. The OTP module is made for the MIDP 2.0 and CLDC 1.0. That is a configuration supported by most new phones today. In addition to MIDP 2.0 and CLDC 1.0 the external package JSR120 must be supported. This allows the MIDlet to send and receive messages as described in chapter 2.6.2.

A MIDlet consists of a Java ARchive (JAR) file containing the source code and a java application descriptor (JAD) file describing the MIDlet properties. Below the JAD file for the OTP module is included. In addition to the required fields some user defined fields must be set to allow the MIDlet to make network connections and register in the push registry as described in 2.6.3.

MIDlet configuration

MIDlet-Permissions: tell which connection interfaces the MIDlet is allowed to use. This MIDlet is allowed to send and receive SMS messages, act as a Bluetooth client and register for the push registry.

MIDlet-Push-1: Creates a push registration that reacts on incoming SMS messages on port 50001.

SMS-Port: Set the port which the MIDlet will listen to. The MIDlet retrieves the SMS-Port by calling the method getAppProperty("SMS-Port").

MIDlet-1:	Midlet,,client.ClientMidlet
MIDlet-Jar-Size:	39784
MIDlet-Jar-URL:	OTPClient.jar
MIDlet-Name:	OTPClient Midlet Suite
MIDlet-Permissions:	javax.microedition.io.Connector.sms, javax.microedition.io.PushRegistry, javax.wireless.messaging.sms.receive, javax.wireless.messaging.sms.send, javax.microedition.io.Connector.bluetooth.client
MIDlet-Push-1:	sms://:50003, client.ClientMidlet, *
MIDlet-Vendor:	Midlet Suite Vendor
MIDlet-Version:	1.0.13
Manifest-Version:	1.0
MicroEdition-Configuration:	CLDC-1.0
MicroEdition-Profile:	MIDP-2.0
SMS-Port:	50003

MIDlet download

As described earlier the OTP Module must be downloaded and installed on the user's mobile phone. When the user has registered he is given access to a site where the MIDlet can be downloaded from. There are two ways to download the MIDlet. The MIDlet can be downloaded to the computer and then transferred to the phone by cable, Bluetooth or IR using the software following the phone. A better option is to use WAP push messages with the URL to the .jar file. Then the mobile phone will download and install the MIDlet automatically. This requires that the mobile phone supports GPRS.

6.2.2 SMS communication

The MIDlet exchange both key exchange massages and authentication messages with the Authenticator over SMS. The MIDlet can send and receive SMS messages by using WMA as described in chapter 2.6.2. By combining WMA with the push registry a MIDlet can be automatically launched upon receiving a SMS. This makes a powerful combination since the MIDlet is started exactly when needed. When the MIDlet is launched it can get the contents of the SMS and process it accordingly. WMA also allows MIDlets to send messages. Therefore the MIDlet can automatically send a response to the Authenticator hiding the whole process from the user.

6.2.3 Key Exchange

The algorithm used for the key exchange is described in section 4.3.2.

6.2.4 Storage of shared secret

The secret key must be stored in the mobile phone memory. The MIDlet stores data in Java ME's Record Store[40]. Storing secrets is always a sensitive point in security. To protect the secret as good as possible the shared secret is stored encrypted. AES is used for encryption and an expanded version of the user password is used as the encryption key. The Record Store is used in private mode to deny any other application from getting access to the records. To be able to check that the password entered at startup is the correct one some version of the user password must also be stored in the record store. Since the password is used for encryption it should not be stored in plaintext. Instead a hash of the password is stored. When the user has entered his password the MIDlet generates a hash of this password and compares it to the hash stored in the record store. If the hash matches, the MIDlet knows that the password is correct.

6.2.5 Bouncy Castle

Bouncy Castle provides a lightweight cryptographic API which can be run on any java platform including Java ME. Form SUN the security functionality for Java ME is found in the additional packages Security JSR 219 and SATSA JSR 177. By using Bouncy Castle a wider security library is available and support for JSR 219 and JSR 177 is not necessary. Therefore the prototype uses the Java ME implementation of Bouncy Castle for adding the cryptographic functionality. For encryption and decryption the AESEngine is used. It is an implementation of the AES (Rijndael), from FIPS 197 [12]. SHA1Digest is used for the hash function. It is an implementation of SHA-1 as outlined in Handbook of applied cryptography[41]. For the message authentication code an HMAC implementation based on RFC2104 [11], is used.

6.3 Server proxy

The server proxy is realized as a Java Applet. When the user starts the authentication procedure the applet is started in the browser on the user’s computer. The applet acts as a Bluetooth server which the MIDlet can connect to. The Bluetooth server and MIDlet use a common predefined UUID value which ensures that the MIDlet connects to the correct device. When the MIDlet has setup a connection with the applet the OTP is transferred using an RFCOMM connection.

By default applets do not have access to local system resources such as network resources. To set up a Bluetooth connection an applet needs access to these resources. This can be solved by signing the Applet. A signed applet is allowed to access system resources defined in a local security policy.

There exist several Bluetooth packages implementing the Bluetooth API. The most common are shown in Table 21.

Table 21. Java Bluetooth APIs

Company Name	javax.bluetooth Support	javax.obex Support	Java Platforms	Operating Systems	Price
Atinav	Yes	Yes	JAVA ME, J2SE	Win-32, Linux, Pocket PC	Unspecified
Avetana	Yes	Yes	J2SE	Win-32, Mac OS X, Linux, Pocket PC	€25, free**
Blue Cove	Yes	No	J2SE	WinXP SP2	Free
Electric Blue	Yes	Yes	J2SE	WinXP SP2	$15 USD
Harald	No	No	any platform that supports javax.comm	Many	Free
JavaBluetooth.org	Yes	No	any platform that supports javax.comm	Many	Free
Rococo	Yes	Yes	JAVA ME, J2SE	Linux, Palm OS	€2500 , free***

This prototype is tested with the BlueCove implementation. BlueCove is a free Java library for the Bluetooth (JSR-82) implementation that currently interfaces with the Microsoft Bluetooth stack found in Windows XP SP2 and Windows Vista. It was originally developed by Intel Research and is currently maintained by volunteers. BlueCove is available for download at http://sourceforge.net/projects/bluecove.

6.4 Authenticator

The Authenticator is realized as a Java servlet running on an application server like Tomcat. Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs.

SMS gateway

To be able to communicate with the MIDlet the Authenticator is connected to a SMS-gateway. The SMS gateway enables the authenticator to send and receive SMS messages just as a regular mobile phone. The SMS gateway can define custom values in the user data header (UDH) of the SMS. By setting this field the SMS will be delivered to the port specified in the header. An SMS that is to be delivered to port 1000, 0x2170 in hex, can have a UDH looking like 06050421700000.

6.5 OTP plug-in

Both Free Radius and Navi Radius (now VitalAAA from Alcatel/Lucent) are radius servers that allow external plug-ins to be written for them. A plug-in is therefore used to generate the specific OTP and key parameters required for this solution. The plug-in will handle generation of the authentication triplet (challenge, otp, mac) and the key exchange parameters prime, private key and public key.

The prime must be of the form 2p+1 where p is also a prime. In addition to this the size of the prime should be considerably large. For Diffie Hellman 512 or 1024 bit key size is recommended. The generation of the keys is described in 4.3.2.

The challenge is generated as a random number using the secure Random function in Java. In the prototype a 20 byte long value is used.

The OTP is simply a hash of a challenge and a secure cryptographic key. The challenge and key are concatenated and then fed into a hash function. The SHA-1 secure hash function produces a 160 bit or 20 byte long hash which is used directly as the OTP.

OTP = hash (challenge || shared secret)

The message authentication code is used to authenticate the server. The MAC is calculated over the OTP and sent to the client together with the challenge. The MAC is produced by feeding the OTP into the HMAC function together with the secret key. This produces a 20 byte MAC.

The challenge and MAC together makes out a 40 byte long message. Encoded as a hexadecimal string this ends up as an 80 character long string which fits easily into a single short message.

6.6 Testing of the prototype

The prototype has been tested according to the use cases presented in chapter 4 to demonstrate that all the described scenarios can be completed.

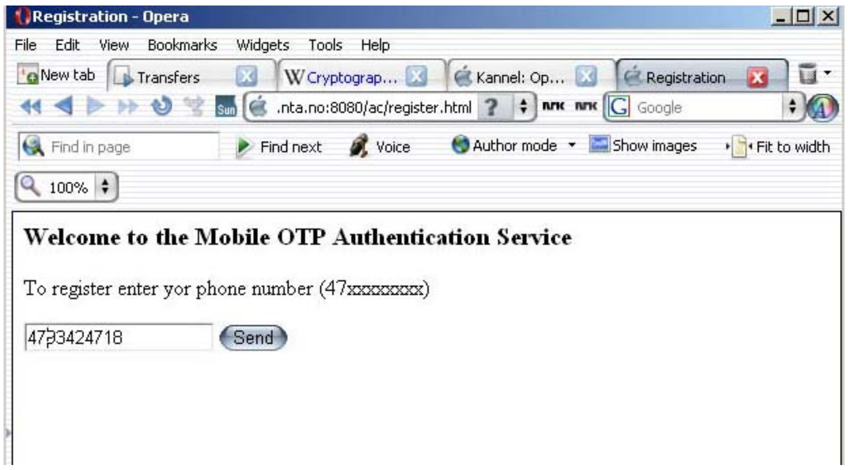
6.6.1 Testing environment

The components that are tested are the MIDlet and the Authenticator. Due to time constraints the server proxy was not integrated in the prototype and was only tested separately. The MIDlet was tested on a Nokia N90 provided by Telenor. The Authenticator was located at the Telenor IP network on xmmap.nta.no and was running on an Apache Tomcat[1] server. The Authenticator uses a Commons HTTP client from the Apache Jakarta Project[2] to connect to the SMS gateway. The gateway is a Kannel SMS gateway[3] and is also located at Telenor. A CGI script is used as a proxy between the Authenticator and the SMS gateway.

6.6.2 Testing scenario

First a new user registers to the server by submitting his phone number on the registration page. Figure 44 shows a screenshot of the registration page.

[1] See http://tomcat.apache.org/
[2] See http://jakarta.apache.org/commons/httpclient/
[3] See http://www.kannel.org/

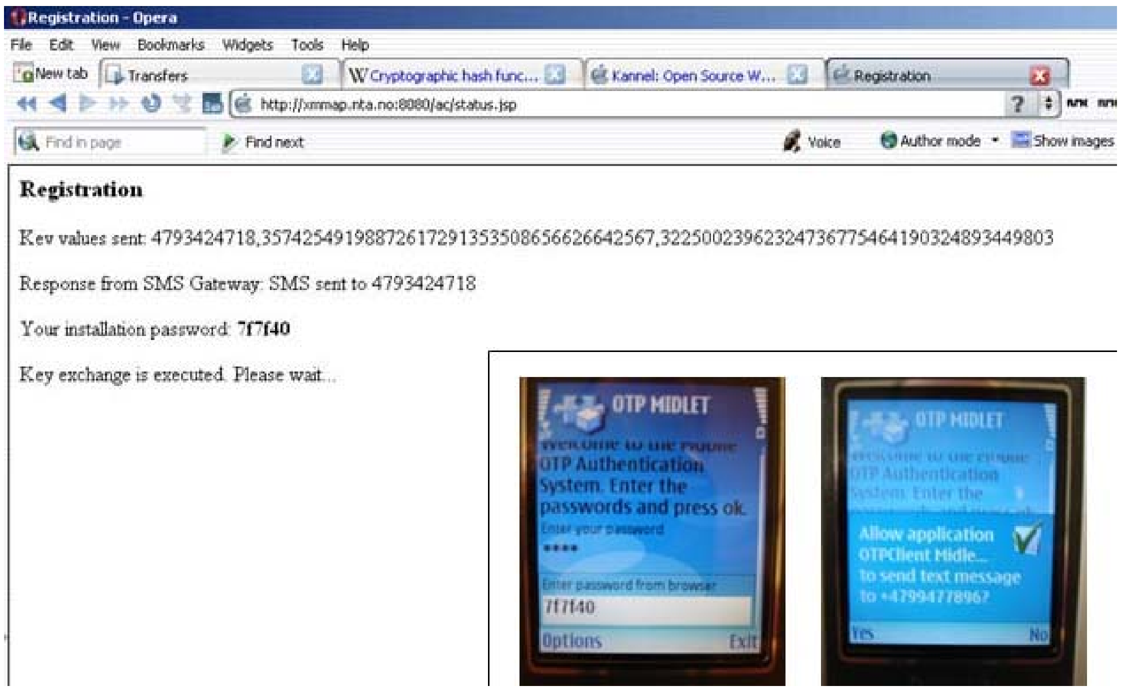

Welcome to the Mobile OTP Authentication Service

To register enter yor phone number (47xxxxxxxx)

4793424718 Send

Figure 44. Screenshot – Register user

When the registration is finished the key exchange started. Figure 45 shows a screenshot from the exchange. For testing purposes the values that are sent are shown in the browser. In the corner screenshots of the mobile phone are shown. The user enters a personal password and the password is shown in the browser. When this is done the MIDlet calculates its key vales and asks the user for permission to send a message back to the Authenticator.


Registration

Kev values sent: 4793424718,357425491988726172913535086566266425 67,32250023962324736775464190324893449803

Response from SMS Gateway: SMS sent to 4793424718

Your installation password: **7f7f40**

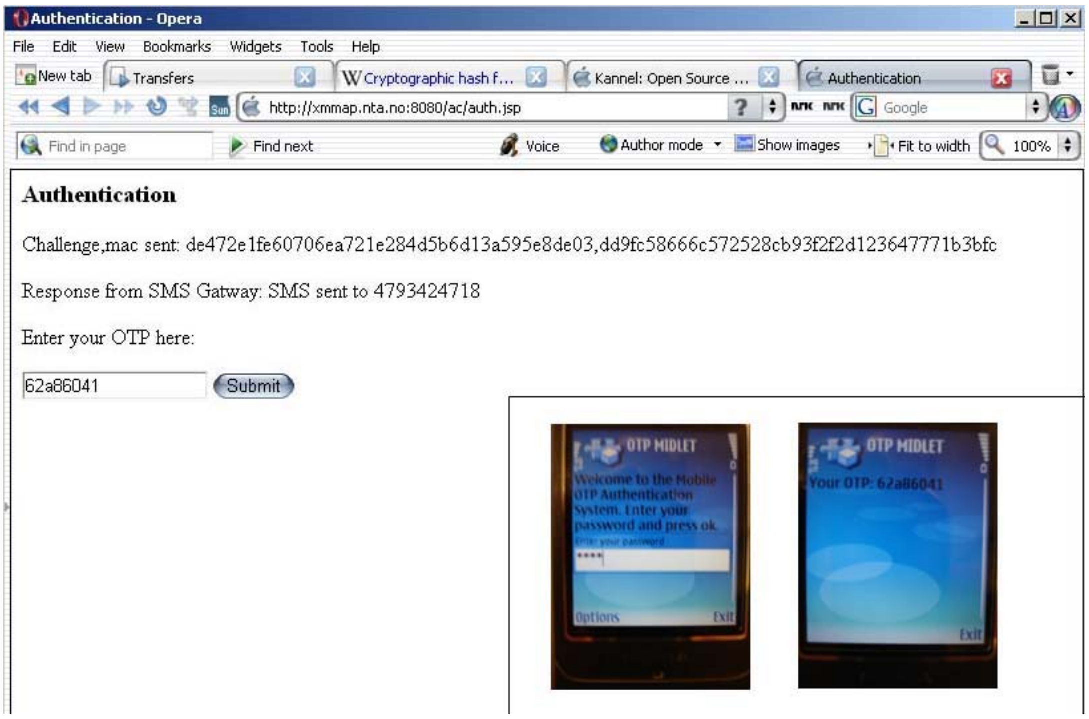
Key exchange is executed. Please wait...

OTP MIDLET
Welcome to the mobile OTP Authentication System. Enter the passwords and press ok.
Enter your password
****
Enter password from browser
7f7f40

OTP MIDLET
Allow application OTPClient Midle... to send text message to +4799477896?

Figure 45. Screenshot - Key exchange

When the key exchange is complete the authentication procedure starts. Figure 46 shows screenshots from the authentication. Since the proxy server was not integrated a manual version is shown where the user has to enter the OPT in the browser. On the mobile phone the user has to enter his personal password. The MIDlet calculates the OTP and displays it on the screen if the MAC is correct.



Authentication

Challenge,mac sent: de472e1fe60706ea721e284d5b6d13a595e8de03,dd9fc58666c572528cb93f2f2d123647771b3bfc

Response from SMS Gatway: SMS sent to 4793424718

Enter your OTP here:

62a86041

Welcome to the Mobile OTP Authentication System. Enter your password and press ok

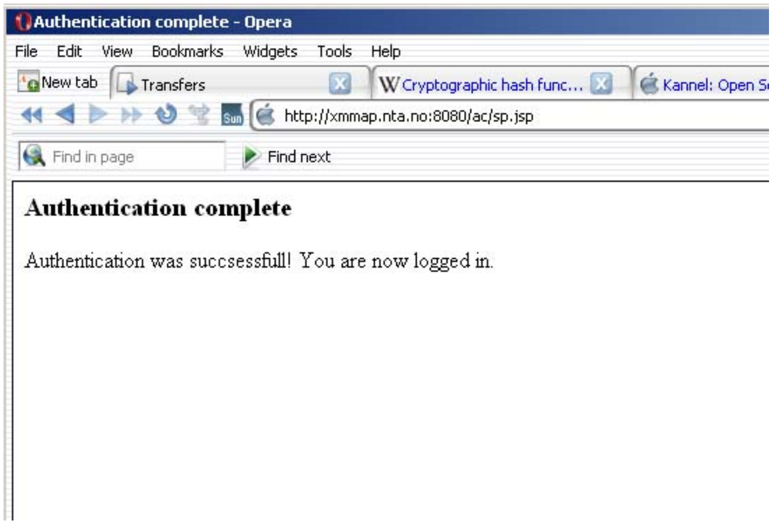
Enter your password

****

Your OTP: 62a86041

Figure 46. Screenshot - Authentication

The user enters the OTP from the mobile in the browser and if the OTP is correct the user is authenticated as shown in Figure 47.


Authentication complete - Opera
File Edit View Bookmarks Widgets Tools Help
New tab
Transfers
Cryptographic hash func...
Kannel: Open S
http://xmmap.nta.no:8080/ac/sp.jsp
Find in page
Find next

Authentication complete

Authentication was succsessfull! You are now logged in.

Figure 47. Screenshot - Authentication complete

6.7 Summary

In this chapter the prototype implementation of the solution is presented. A possible deployment is shown and the implementation of the different components is described. Finally a demonstration of the prototype is described, showing the most important functions of the solution.

7 Security evaluation

The security analysis will further elaborate the security risks and properties in the system. The evaluation follows a stepwise approach based on a model from Common Criteria [42].

1. Identify the security environment subject to the evaluation.
2. Define a threat model describing the risks and threats the system is subject to.
3. Describe the security objectives of the system and how and which threats they counter.

In addition the evaluation contains a description of possible attacks and the feasibility of these attacks.

7.1 Security environment

The evaluation will focus on the components and protocols used and developed in this thesis. The communication and authentication between the client and Authenticator is therefore the main focus of this evaluation.

7.2 Threat model

The threat model describes attacks and threats which the security environment is subject to. First possible threats are described and then a risk evaluation of the different components a presented.

7.2.1 Threats

Eavesdropping - An attacker can try to listen in on communication between the client and authenticator to learn sensitive information.

Dictionary attack – Instead of trying all possible combination when guessing a password, the attacker only tries with password from a dictionary. The dictionary contains passwords which is most similar to what the victim will use. The dictionary attack is effective since people tend to use single words found in a dictionary or variations which are easy to predict.

Masquerade - An attacker pretends to be someone he is not. The attacker might masquerade both as client and authenticator. Either by replaying a valid authentication sequence or setting up a false authentication service to try to lure the client to give up secret information.

Replay attack - An attacker tries to reuse an old authentication sequence to gain unauthorized access

Modification - An attacker alters a message or the contents of a message to produce an unauthorized effect

Session hijacking - A valid session that is already established is taken over by an attacker which will then get unauthorized access.

Phishing - An attacker uses social engineering techniques to try to get the user to give up secret information.

Man-in-the-middle (MITM) attack - An attacker can intercept, read and modify all messages going between the client and authenticator without any notice.

7.2.2 Component risk evaluation

Figure 48 shows the components of the security environment. Below follows a risk evaluation of each component.

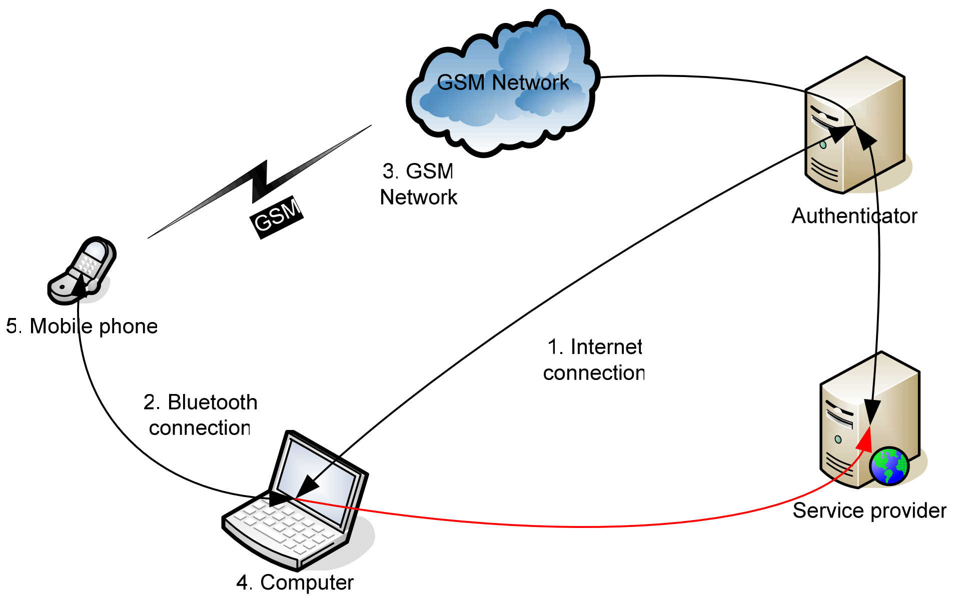


Figure 48. Component threat model

1. Internet connection

The Internet connection between the client and Authenticator is the most vulnerable part of the system. An open Internet connection is open to anyone with relevant computer skills and is subject to all the attacks described above.

2. Bluetooth connection

Any network connection is in principle open to the same kind of attacks. Due to the short range of the Bluetooth technology the number of possible attackers on the Bluetooth link is reduced dramatically. Even so an attacker can mount any of the attacks listed above if he is in range of the Bluetooth device.

3. GSM network

The GSM network has well defined security mechanism deflecting most of the attacks described above. Even so there are some weaknesses in the system that can be exploited in an attack. Since there are no authentication of the network in GSM the system is vulnerable to a MITM attack. How such an attack can be done is described in [43].

4. Computer

Malware such as trojans and worms can infect the user's computer and be used to get sensitive information stored on the computer.

5. Mobile phone

An attacker can steal the mobile phone and use this to masquerade as an authentic user. The attacker might also try to tamper with a mobile phone in order to obtain secret information such as the shared secret.

7.3 Security objectives

This section presents the security objectives of the solution. It is divided into the systems main three parts registration, key exchange and authentication. Each part contains a description of how they are secured and how these security objectives counter the possible attacks the part is subject to.

7.3.1 Registration

During registration the user gives some personal information about him self. None of this information is necessary secret, but to avoid information to be altered or given to the wrong people the link should be secure. Authentication of the user is not required or possible at this point since the user is new to the system. Authentication of the server on the other hand is important so the user knows that he registers to a valid service.

When registration is complete the server generates a password which is used to authenticate the following key exchange. This password is presented to the user in the browser. This password must be protected and must therefore be sent over a secured link.

Transport Layer Security (TLS)

The solution uses TLS to secure the registration. As described in 2.2.1 the protocol uses public-key cryptography for authentication and to exchange strong cryptographic session keys for message encryption and authentication. After some initial messages the server sends its certificate to the client. The certificate is signed by a trusted third party so the user can verify that the server is authentic. Then the client generates session keys and uses the server’s public key to encrypt them before sending them to the server. In this way the server is authenticated and the client and server have securely exchanged session keys. The session keys are used to encrypt all subsequent communication between the client and server resulting in a secure session.

Through authentication of the server possible phishing attacks are avoided. With message encryption eavesdropping, message modification and session hijacking are efficiently countered.

In some early versions of SSL a 40 bit key where used for encryption due to export restrictions. A 40 bit encryption key is not strong with the computer power available today. In TLS the keys are 128 bit or more which is the standard for safe encryption.

Some potential security holes have been pointed out in TLS. A new version TLS 1.1 is released addressing several of these problems. Especially an attack discovered by Daniel Bleichenbacher which can be launched against TLS 1.0 servers and recommendations against timing attacks are addressed [44]. TLS is the best choice for securing web application communication today and is considered very safe.

7.3.2 Key Exchange

The key exchange is an authenticated version of the Diffie-Hellman key exchange. Diffie-Hellman was constructed to securely exchange a secret key between two parties over an open network. The problem with Diffie-Hellman is that the communicating parties are not authenticated to each other and is therefore vulnerable to a MITM attack. By authenticating the key exchange with a password such an attack is countered.

The key exchange is preformed by SMS. These messages are therefore protected by the security mechanisms in GSM. As described earlier there is possible to mount a MITM attack on GSM. Even though an attacker manages to mount such an attack the key exchange is secure. The attacker can observe the messages sent by the client and Authenticator, but will not be able to deduct the key.

Secret key management

The secret key is stored on the mobile phone using Java record store. The record store is used in private mode denying any other application access to the records. In addition the secret key is encrypted using AES before it is stored. A padded version of the personal password is used as the encryption key. In addition to act as an encryption key the personal password also ensures two-factor authentication. Each time the MIDlet is started is checks that the correct password is entered. To be able to do this without storing the password in clear-text a hash of the password is stored instead.

The secret key and personal password are never transmitted over a communication link. The encryption of the secret key and use of a password protects against attack directed at the mobile phone. The password prevents stolen mobile phones from being misused. An attacker trying to get authenticated by a stolen phone will get know where without also getting hold of the password.

Bogus applications secretly installed on a phone by an attacker are not allowed access to the records because of the private mode. An attacker might get to the records by stealing a mobile phone and read the whole internal memory. Since the records are encrypted before storing this would not give the attacker access to the key.

Advanced Encryption Standard (AES)
AES is the NIST standard for symmetric encryption and In June 2003, the US Government announced that AES may be used for classified information:
"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use."[45].

As of today the only successful attacks on AES are side channel attack. To ensure that no implementation errors occur a clean implementation of AES as described in [12] is used.

7.3.3 Authentication

For authentication a challenge and MAC is sent to the client by SMS. An attacker that has mounted a MITM attack on the GSM system can pick up the message and read its contents. The challenge is a random number giving no information to the attacker. The MAC is calculated over the OTP corresponding to the challenge. This MAC is made using the HMAC algorithm. At present no known attack is possible against HMAC. Since HMAC produces a 160 bit hash the attacker must recover 2^80 messages before he can learn anything useful from such an attack.

The client calculates the OTP from the challenge received from the server and the secret key. A MAC of the OTP is also calculated to check against the MAC received from the server. By doing so the client can verify that the challenge is coming from an authentic server. Therefore both the server and client are authenticated during authentication. This mutual authentication prevents MITM and phishing attacks. This also prevents an attacker from being able to generate fake challenges to produce wanted output for in an attempt to figuring out the key. Only if the challenge is authentic the client sends the OTP back to the server over Bluetooth and Internet. Else an error message is sent back canceling the authentication.

OTP are not vulnerable to dictionary and replay attacks. OTP appear as random sequences rendering a dictionary attack useless. As the name implies OTP can only be used once and therefore replaying an OTP will only result in the system noticing that it is under attack. Session hijacking is avoided by securing the link between the client and Authenticator with TLS.

Bluetooth connection

When setting up a Bluetooth connection one can choose to enable encryption or not. The OTP is secured by its properties, but to ensure that the OTP is coming form a trusted device the connection requires encryption. The encryption keys are based on a passkey chosen by the user during pairing. To ensure strong encryption keys of 128 bit a 16 digit long passkey is required.

Secure Hash Algorithm 1 (SHA-1)

SHA-1 is used for OTP generation. A brute force attack on such an algorithm takes 2^{80} trials to find a collision since SHA-1 produces 160 bit output. In 2005 three Chinese described an attack that found collisions in SHA-1 in 2^69 trials [46]. This is still an immense task, but the security of SHA-1 was questioned. Using a SHA implementation that produces a slightly longer hash, like SHA-224, solves this problem and has no impact on this solution.

7.4 *Possible attacks*

This section describes three different attacks which can be launched against the solution.

7.4.1 Brute force attacks

Brute force attacks are attacks where an attacker user exhaustive techniques to break a system. In this case the most obvious attack is for an attacker to gather challenges and corresponding OTP and from this try to guess the key by feeding random numbers into a hash function together with the challenge and then continue until the output matches the OPT. Using a 160 bit long key would leave the attacker with at 2^160 possible passwords to guess from.

Another brute force attack would be to try to guess the OTP. Since the password is sent automatically back to the authenticator user mistakes in typing the password is avoided and only one attempt in presenting the password is allowed. If the password is incorrect the whole authentication process will have to be repeated. A timing factor controls the frequency of authentication attempts from the same client. The factor increases with the number of consecutive tries slowing down such an attack considerably. With a password length of 160 bit this becomes an impossible task.

7.4.2 Combined attack

Combining several different attacks might lead to a possible attack on the scheme. If an attacker manages to mount a MITM attack on the GSM system he can intercept the SMS messages sent during the key exchange. If the same attacker also has been able to place a trojan horse in the victim’s computer, the trojan can possibly get the password which is used to authenticate the key exchange. The trojan must be able to get and send the password to the attacker before the key exchange is started which will say in within 30 seconds. Then the attacker can perform a key exchange with the server acting as the victim and get access to the secret key. If this succeeds the attacker can manage to get authenticated using the victim’s identity.

Mounting such an attack requires big resources. Mounting a MITM attack on the GSM system is far from trivial. The attacker must be able to set up a fake base station nearer to the user than any other authentic base station so the user’s phone will connect to the fake station. In addition the attacker must have managed to infest the user’s computer with a trojan which is able to get the password from the browser and send it to the attacker in real-time.

7.4.3 Offline attack

Concerns have been voiced about storing secret information in the internal memory of a mobile phone since this is not considered to be tamper proof. An attacker can potentially be able to scan the contents of the memory and in this way get access to the secret information. The solution counters this attack by encrypting the secret information before storing it. The data is encrypted with an expanded version of a user chosen password as the encryption key. For usability considerations this password can not be to long and the encryption key will therefore be a weak point. Cracking this key must therefore be considered quite possible.

At first glance this seems like an attack which can drastically reduce the security of the solution, but a closer analysis of the attack reveals quite e few obstacles for the attacker.

First the attacker must be able to scan the mobile phones memory without the owner noticing it. If the owner notices that the phone is stolen he will notify the authentication authority which will discard the key.

Second the attacker must be able to identify where in the phones memory the key is stored. How to do this is not clear since the mobile phones constantly keep changing things around to store data efficiently and the encrypted key has no common pattern which the attacker can search for.

Given that the attacker manages to locate the key he would need some way to check if he managed to decrypt the key correctly. This can be done with a challenge/OTP pair. To obtain such a pair the attacker must crack the security of the GSM system to obtain the challenge and then crack the encryption of the client-authenticator link to obtain the OTP.

Taking all these challenges into account the attack ends up with being very hard to accomplish and requiring big resources.

7.5 Summary

Protecting the user equipment against malware is beyond the scope of an authentication scheme. Even though when constructing a security solution all threats should be considered, full protection against careless is impossible to achieve as long as the user is involved in some way.

Brute force attacks are always possible to perform against security solutions. If these are the best possible attacks against a scheme the key size decides the strength of solution. Today key lengths of 128 bit or longer are considered impossible to break in real time for symmetric encryption.

Even though the offline attack turned out to be a very complicated attack the concern about storing secret information on non-tamper proof media is just. Therefore storing the secret on for example the SIM card or other tamper proof media should be considered in similar solutions.

The scheme depends on TLS for securing the communication between the client and Authenticator. Another possible solution would be to use the shared secret to generate session keys for encryption and message authentication of further communication. This would increase the necessity of considering other ways to store the key as commented above.

8 Conclusion

8.1 Contributions

This thesis has presented a novel secure authentication solution based on the OTP principle. Through the study of several authentication schemes based on the mobile phone and its SIM card a solution with the mobile phone acting as an OTP token where chosen to be further elaborated.

An analysis and a design of the solution have been done and a prototype implementing the main functionality has been developed. Finally a security evaluation of the solution is included identifying threats, security objectives and possible attacks. The prototype has been implemented in java using the different java libraries available for the different devices. The Java ME library was used to develop a MIDlet for the mobile phone turning it into an OTP token. A Java applet was developed to act as a Bluetooth server running on the user's computer automatically relaying the OTP from the mobile phone to the Authenticator. The Authenticator is realized as a Java servlet which can run on any application server such as the Apache Tomcat. The Authenticator handles the client communication and transfers authentication messages between the client and Authentication Server.

The proposed solution offers a strong authentication scheme which can substitute many of the authentication schemes we are using to day. Not only can it replace the standard username/password scheme, but due to its security services it can also replace stronger schemes such as existing OTP and smartcard solutions. Therefore the solution is suitable for many services on the Internet which requires authentication such as Internet banking, corporate intranet, Internet stores and e-Government applications.

During the work with the thesis the author wrote a paper "Using the mobile phone as a security token for unified authentication". The paper describes the same solution as this thesis and was accepted to the IARIA conference ICSNC 2007. The paper is shown in the appendix.

8.2 Critical review

Usability

In addition to creating a secure authentication scheme the usability was also an important issue to improve compared to existing solutions. By using the mobile phone as the authentication token no extra hardware is required on the client side and since most people always carry their mobile phone with them the authentication token is always available. As said earlier the solution can be used towards almost any service on the Internet requiring authentication. The user can therefore use this solution towards all his services and therefore only needs to relate to one authentication mechanism for electronic authentication.

By using SMS as a delivery protocol a small authentication delay is introduced. However, the delay is in the same range as in existing solutions and does not cause any reduction in usability.

Deployment time and cost

In several of the solutions described in chapter 3 a new SIM card is required. The solution which is described in the rest of the thesis uses the SIM card as it is. This dramatically reduces the deployment time of the solution. Receiving a new SIM card usually take days or even weeks, while a new user to this solution can get authenticated within minutes from the first access request. Avoiding having to replace the SIM card also reduces the deployment cost for both credential and service providers.

Availability, reliability and scalability

If the authentication system is down the user will not get access to the wanted service, even if the service is available. It is therefore important that the authentication has a high availability and reliability. The GSM system is considered to be very reliable and it is the most deployed mobile network in the world. The simplicity of the solution reduces possible error causes and makes it easy to implement error corrections.

It is important that the system scale well when the number of users increase. The system can easily be distributed both physically and geographically to avoid possible bottlenecks.

Possible improvements

In the summary of the security evaluation some security considerations about the solution were presented. The most important being the storing of the secret key in the internal memory of the mobile phone. Storing the secret on tamper proof media should therefore be considered, especially if the secret key shall be used to generate session keys for encrypting further communication.

Another consideration regards the key exchange. A key validation might be necessary to ensure that both parties have generated the same key. If an undetected error occurs during key exchange so the same key is not generated authentication will fail in the next step.

8.3 Future work

The prototype developed in this thesis only implements the main functionality for proving the concept of the solution. For commercial use a full implementation should be developed and tested with all the security features implemented including the improvements described above. The solution should also be properly integrated with an identity management platform an incorporated into a single sign-on (SSO) environment.

Storing the shared secret on the SIM card or on other tamper proof media will further increase the security and is an interesting field for further study. As the interface toward the SIM card become more standardized and phones supporting this become available on the market this should be explored.

References

1. Hallsteinsen, S., *A study of user authentication using mobile phone*, in *Department of Telematics*. 2006, Norwegian University of Science and Technology: Trondheim.
2. FreeAuthProject. *The FreeAuth Project*. [cited 2007 March]; Available from: http://www.freeauth.org/site.
3. mOTP. *Mobile One Time Passwords*. [cited 2007 11th January]; Available from: http://motp.sourceforge.net/.
4. NordicEdge. *NordicEdge*. [cited 2007 March]; Available from: http://www.nordicedge.se/.
5. OATH. *Initiative for open authentication* [cited 2007 February]; Available from: http://www.openauthentication.org/.
6. Burr, W.E., D.F. Dodson, and W.T. Polk, *Electronic authentication guideline* 2006.
7. Stallings, W., *Network security essentials*. 2003: Prentice Hall.
8. Thanh, D.v., et al., *Offering SIM Strong Authentication to Internet Services*. 2006.
9. Haller, N., et al., *A One-Time Password System*. 1998, IETF.
10. NIST, *FIPS PUB 180-1 Secure Hash Standard*. 1995.
11. Krawczyk, H., M. Bellare, and R. Canetti, *RFC 2104 HMAC: Keyed-Hashing for Message Authentication*. 1997.
12. NIST, *FIPS 197 - Advanced Encryption Standard (AES)* 2001.
13. Smith, C. and D. Collins, *3G wireless networks*. 2002.
14. NetworkDictionary. *GSM: Global System for Mobile Communications*. 2006 [cited 2006 06/09]; Available from: http://www.networkdictionary.com/wireless/gsm.php.
15. Wong, K.D., *Wireless Internet telecommunications*. 2005, Boston: Artech House.
16. Quirke, J. *Security in the GSM system*. 2004 [cited 2006 27/05]; Available from: http://www.ausmobile.com/downloads/technical/Security%20in%20the%20GSM%20system%2001052004.pdf.
17. Ghosh, S. *Extend J2ME to Wireless Messaging*. 2003 [cited 2007 May]; Available from: http://www-128.ibm.com/developerworks/wireless/library/wi-extendj2me/#figure1.
18. Aboba, B., *RFC 3748 - Extensible Authentication Protocol*. IETF, 2004.
19. H. Haverinen, J.S., *Extensible Authentication Protocol for GMS-SIM*. IETF, 2006.
20. Edney, J. and W.A. Arbaugh, *Real 802.11 security: Wi-Fi protected access and 802.11i*. 2004, Boston, MA: Addison-Wesley.

21. Gehrmann, C., J. Persson, and B. Smeets, *Bluetooth security*. 2004, Boston: Artech House.
22. Bluetooth-SIG, *Bluetooth Core Specification v.2.0*. 2004.
23. Bluetooth-SIG, *Bluetooth specification SIM Access Profile v.10*. 2005.
24. ETSI, *GSM 11.14 Specification of the SIM Application Toolkit for the Subscriber Module - Mobile Equipment (SIM - ME) Interface*. 1996.
25. IBM. *SIM access profile*. [cited 2006 17/11]; Available from: http://www-128.ibm.com/developerworks/wireless/library/wi-simacc/.
26. SUN-Microsystems. *Java ME Technologies*. 2006 [cited 2006 18/11]; Available from: http://java.sun.com/javame/technologies/index.jsp.
27. Cricco, R. and Y. Vinson, *Integrating the SIM Card into J2ME as a Security Element* 2005.
28. Ortiz, C.E. *The Wireless Messaging API*. 2002 [cited 2007 March]; Available from: http://developers.sun.com/techtopics/mobility/midp/articles/wma/.
29. Ortiz, E. *The MIDP 2.0 Push Registry*. 2003 [cited May 9th 2007]; Available from: http://developers.sun.com/techtopics/mobility/midp/articles/pushreg/.
30. Roy, S. *Push messages that automatically launch a Java mobile application*. 2006 April 17th [cited 2007 March].
31. Motorola. *Java APIs for Bluetooth(JSR 82)*. 2005 [cited; Available from: http://jcp.org/aboutJava/communityprocess/mrel/jsr082/index.html.
32. GSM Mobile, N.Z. *SIM Toolkit*. [cited 2006 17/11]; Available from: http://www.gsmmobile.co.nz/Sim_Toolkit.htm.
33. Lars Lunde, A.W., *Using SIM for strong end-to-end application authentication*, in *Telematics*. 2006, NTNU: Trondheim.
34. 3GPP, *3GPP TS 23.040, Technical realization of the Short Message Service*. 2003.
35. Malan, R. and D. Bredemeyer, *Functional Requirements and Use Cases*. 2001.
36. Malan, R. and D. Bredemeyer, *Defining Non-Functional Requirements*. 2001.
37. Fowler, M., *UML distilled: a brief guide to the standard object modeling language*. 2004, Boston, Mass.: Addison-Wesley.
38. Jablon, D., *Strong Password-Only Authenticated Key Exchange*. Computer Communication Review, ACM SIGCOMM, 1996. **vol. 26**(no. 5).
39. Rescorla, E., *RFC 2631 Diffie-Hellman Key Agreement Method*. 1999.
40. Microsystems, S. *Java ME Technology APIs & Docs*. [cited 2007 March]; Available from: http://java.sun.com/javame/reference/apis.jsp.

41. Menezes, A.J., P.C. Van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*. 1997, Boca Raton: CRC Press.
42. CommonCriteria, *Part 1: Introduction and general model.* Common Criteria for Information Technology Security Evaluation, 2006.
43. U.Meyer and S.Wetzel, *A Man-in-the-Middle Attack on UMTS.* In Proceedings of the 2004 ACM Workshop on Wireless Security, 2004.
44. Dierks, T. and E. Rescorla, *RFC 4346 - The Transport Layer Security (TLS) Protocol Version 1.1*. 2006.
45. CNSS, *National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information*, in *CNSS Policy No. 15, Fact Sheet No. 1*. 2003.
46. Wang, X., Y.L. Yin, and H. Yu, *Finding Collisions in the Full SHA-1*. 2005.

Appendix A Attachments

The source code for the prototype is attached as a ZIP file located in DAIM. The attachment contains:

Source code for the prototype
Install ready JAR and JAD files for the MIDlet suite
Servlet including web.xml, jsp and html files
CGI script
A “read me” file explaining how to launch the prototype

Appendix B Accepted Paper to ICSNC 2007

The paper “Using the mobile phone as a security token for unified authentication” was accepted for the IARIA (International Academy, Research, and Industry Association) conference The Second International Conference on Systems and Networks Communications (ICSNC 2007).

The email announcing the acceptance of the paper is presented below. The final version of the paper is included at the end.

```
Dear Steffen Hallsteinsen,

On behalf of the Program Committee, we are happy to inform you that your
paper 48 ("Using the mobile phone as a security token for unified
authentication") has been accepted for publication and presentation at
ICSNC'07 and Workshops.

The acceptance of your paper is made with the understanding that each
accepted paper will be registered and at least one author will attend the
conference to present the paper (preferably with PowerPoint slides). 10-14
slide deck is perfect.  Conference rooms will have computers and video
projectors.

Registration: Each accepted paper must be separately registered. The
registration form is available on the conference web site: see
"Registration form". Note that a paper will be published on the IEEE Xplore
and Conference Proceedings only after the paper is registered, i.e., the
registration form is sent in the due time and successfully processed.
Please fax the registration form before uploading the paper and the
copyright transfer form. The hotel booking can be sent later, using the
same registration form. However, as there is a hot season, please book the
conference hotel as soon as possible; on June 1, 2007 our contractors must
release the pre-booked extra rooms.  All the registration related issues
must be addressed to manuela@vicov.com.

The deadline for the submission of registration forms to the organizers is
June 1, 2007. This helps to draft the preliminary program and book the
services. Otherwise, the paper is considered withdrawn and it will neither
appear in the proceedings of the conference nor published on the IEEE
Xplore Digital Library.

Camera ready submission and copyright transfer form: For the camera ready
submission, please consider the reviewers' comments or guidelines when
editing your final version. The Proceedings are published by the
IEEE Computer Society Press. Please read carefully the "Manuscript
Preparation" on the conference web site when submitting the final version
and the copyright transfer form. You can still update the originally and
timely submitted camera ready on the IEEE site later on, as shown in
"Manuscript Preparation" until June 1, 2007. For any technical submission
problem please use the e-mail address of the bottom of the "Manuscript
preparation" web page. Please follow the steps properly. The author is
supposed to receive a PDF receipt when the file has been successfully
```

attached. If the author does not receive the receipt then something has gone amiss.

Thanking once again for your active participation, we are looking forward for your cooperation to successfully finalize the event.

Again, register first, then, please accurately consider the review comments when finalizing your camera ready version. The chairs and the logistics team will check for the conformance with the reviewers' comments.

Please fill the registration form as soon as possible, including hotel booking. Flight information can be sent later, using the same registration form.

Sincerely yours, Events' Chairs ----- Reviews ------

* Comments to Authors The paper describes a method to authenticate via a mobile phone. Related work is not discussed thoroughly, and only some similar methods are mentioned as using one time passwords (OTP) over phones; this makes hard to assess in novelty, as I cannot precisely know the advantages of this paper.

This paper claims to contribute a solution that prevents simple OTP eavesdropping, by transmitting a challenge that is hashed with a secret key to generate the OTP, which is never sent via unsecured network. The generation of the secret key is done via a variant of the (authenticated) diffie-hellman key exchange. This seems sound to me, although I'm not sure how the mutual authentication is done to get this key (the user seems to choose a password online, with no prior authentication at that phase?).

The ''security analysis'' is quite informal, and not every threat as listed by the authors is addressed (are all of them vulnerable?).

Overall, i like the idea of authenticating via a mobile phone, but i'd like the paper to be more thorough in their related work and clear in the specific contributions and claims. The security analysis would benefit from more formality and technical discussion of the merits and security of the approach.

----------------------*----------------------

* Comments to Authors The authors present the use of mobile phones as security tokens as a new idea but actually many researchs have been done on this field and now products can be found.

I suggest to read and include in a related works section: "secure web authentication with mobile phones", "Authentication using multiple communication channels", "INternet ID - Flexible re-use of mobile phone authentication for secure access" or "strong authentication on mobile devices".

Also, the authors explain that users should today remember too many couples login/password, which is true. Many works deal with federation of identity (e.g. Shibbolteh with SAML, WS-security framework, Microsoft NET passports, etc.) but there are not mentionned in this article.

Finally I'm not sure that splitting threats and security properties (actually these are security services, not security properties) in the last section is a good choice. The reader needs to read the list of threats, then the list of security services and the threats again in order to understand which of the threats are handled by the solution.

```
----------------------*-----------------------
```

Using the mobile phone as a security token for unified authentication

Steffen Hallsteinsen
Department of Telematics,
Norwegian University of Science and Technology
Trondheim, Norway
steffeng@stud.ntnu.no

Ivar Jørstad
Ubisafe
Oslo, Norway
ivar@ubisafe.no

Do Van Thanh
Telenor R&I
Oslo, Norway
thanh-van.do@telenor.com

***Abstract* - The number of different identities and credentials used for authentication towards services on the Internet has increased beyond the manageable. Still, the most common authentication scheme is based on usernames and passwords. This is a weak authentication mechanism, which can be broken by eavesdropping on the network connection or by sloppy handling by the users (e.g. re-use of the same password for different services, writing down the passwords on paper etc.). Also, management of user credentials is a costly task for most companies, estimated by IDC to around 200-300USD pr. user/year. Hence, better solutions for simplified, yet secure authentication, is required in the future. This paper proposes and describes an authentication scheme based on a One-Time Password (OTP) MIDlet running on a mobile phone for unified authentication towards any type of service on the Internet.**

Keywords-component; authentcation, mobility

I. INTRODUCTION

More and more services are becoming available on the Internet, and many of these services require authentication. The most widespread solution for electronic authentication today is the use of a username and password. As the number of services grows so does the number of username and password pairs that the user needs to remember. Already many people are experiencing that it is impossible to remember all the combinations and therefore they use the same pairs for all their services and choose passwords that are easy to remember. This strongly reduces the security of an already weak authentication mechanism. Several more secure solutions for electronic authentication exist, such as One-Time-Password (OTP) or Smart Card PKI solutions. They solve the security problem with passwords, but they most often increase the burden for the user. Extra hardware and software is required both for the user and service provider and they are often specific for each service so the user needs to deal with many different devices and procedures.

This paper addresses this problem by proposing an authentication scheme using the mobile phone as an authentication token. The GSM system is already well understood by the user and the proposed solution does not require any extra hardware device at the user side. By using the mobile phone as hardware token, strong authentication can be realized with a system and a device that the user already has and knows how to use.

The mobile phone has several advantages which can be exploited in user authentication:

- most people already have one
- the mobile phone has computing and communication capabilities that allows automation of the authentication process, relieving the user of this burden
- there are already good security mechanisms built into the GSM system that may be exploited.

The paper starts with describing the process of electronic authentication. It continues with explaining the benefits of OTP compared to normal passwords and what is required for using the mobile as an authentication token. Then the mobile OTP solution is described and analyzed in detail.

II. RELATED WORK

The Free Auth Project has made a MIDlet version of their OTP solution [1]. Their OTP generation is based on a time factor and requires that the client and server are synchronized for the solution to work. Time synchronization is not an easy task and the solution is vulnerable if the synchronization fails.

Several solutions exist where an OTP is sent to a mobile phone and used in addition with a static password or other authentication tokens. NordicEdge AB [2] uses this approach in their OTP solution. Even though additional hardware is avoided extra user operations are needed and the OTP is sent two times over the network in clear text.

The solution presented in this paper is automated to keep the user burden at a minimum. The application is automatically launched through push registry and SMS messages and the OTP is automatically transferred back to the server through Bluetooth and Internet. A combined use of the GSM network and a secure Internet connection to exchange the authentication messages results in a secure multi-channel authentication scheme.

III. Authentication process

Authentication is the assurance that the communicating entity is the one that it claims to be [3]. A system can authenticate a user to determine if the user is authorized to perform an electronic transaction or get access to information or a system.

The authentication process begins with a client requesting a service which requires authentication. The client is asked to present a token to prove his identity. A token binds a user's identity together with a secret and is given to the user during registration. When a user presents his token during authentication the authenticating party can verify the user's identity since the token is unique for each user.

Tokens consist of one or more of the following factors:

1. something you know, e.g. passwords
2. something you have, e.g. hardware token
3. something you are, e.g. biometrics

Single-factor authentication such as passwords, which is only based on something you know, is shown to be open for many types of attacks. Eavesdropping, dictionary attacks and replay attacks can all be used to break such a system[4].

Strong authentication schemes rely on more than one factor. This dramatically increases the security. By combining something you know with something you have, an attacker needs to physically steal something in addition to learn the secret. Since the shared secret in such solutions are not sent over the network also learning the secret becomes much harder.

Multi-channel communication is another way to further improve the security of an authentication scheme. By using several communication channels both eavesdropping and man-in-the-middle attacks are much harder to accomplish since the attacker must control all the channels.

As will be shown, authentication using the mobile phone can exploit both these two improvements.

IV. Basis for mobile authentication

To be able to authenticate users through a mobile phone, certain main components must be in place. Fig. 1 shows the main components and the basic architecture needed for the solutions described later to work.

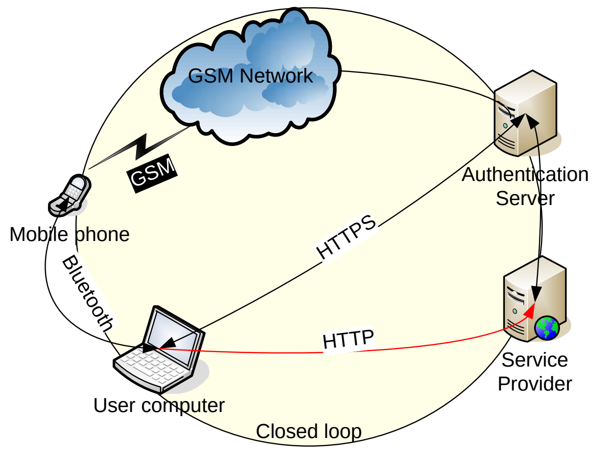


Figure 1. General architecture of mobile authentication

The user must have access to a computer connected to the Internet and be in possession of a mobile phone with a working SIM card. If the computer and mobile phone is equipped with Bluetooth higher usability can be obtained. Through the Internet browser on the computer the user can access web services provided by service providers. The service provider (SP) is connected to an Authentication Server (AS) that will handle the authentication on behalf of the SP. The AS is connected to the GSM network which enables it to communicate with the user's mobile phone and AuC. The AS is composed of two parts, an authenticator and an AAA server. The authenticator communicates with the client and relays messages to the AAA server which handles the authentication.

When designing an authentication scheme which uses two separate devices which communicate over two different networks it is very important to ensure that it is the same user that controls both devices. This is done by ensuring that there is a "closed loop" going through all the components involved in the authentication as illustrated in Fig. 1. The loop starts in the device requesting the service, goes through the network with Service Provider (SP) and Authentication Server (AS) and then via the mobile phone and back to the initial device either by user interaction or Bluetooth.

This closed loop can be realized in several ways. This paper describes how using the mobile phone as an OTP token does this.

V. One-Time Passwords

One-Time Password is one of the simplest and most popular forms of two-factor authentication today. For example, in large enterprises, Virtual Private Network access often requires the use of One-Time Password tokens for remote user authentication [5].

One-Time-Passwords solve many of the problems with user-chosen passwords. They are generated by a computer and can therefore appear totally random. In addition as the name suggests each password is only used once. That makes them invulnerable to replay attack and sniffing. By changing each time a password is needed OTP solutions introduce liveness. This is an important concept in security since it is possible to detect if someone is using old information.

The security of the OTP system is based on the non-invertability of a secure hash function. Such a function must be tractable to compute in the forward direction, but computationally infeasible to invert [6]. This hash function can be deployed in three different ways to generate the OTP. The first type applies the hash function a given number of times to generate a chain of OTPs where the next OTP is generated by taking the hash of the previous one. The second type is based on time-synchronization between the authentication server and the client. The current time is used as seed for the hash. As long as the server and client are synchronized they generate the same OTP. The third type uses a challenge, e.g. a random number, as seed to the hash function.

VI. The mobile OTP solution

The mobile OTP solution combines the simple and secure OTP principle with the ubiquitousness of a GSM mobile phone. A Java MIDlet which can be installed on any Java enabled mobile phone transforms the phone into a secure OTP token which can be used to log in to any service on the Internet.

The solution is based on a simple challenge-response protocol. When the user wants to log in he presents his username, and a challenge is sent to the user's mobile phone. The OTP MIDlet installed on the phone generates an OTP from the challenge and sends it back to the server. The server verifies that the OTP is correct and the user is authenticated.

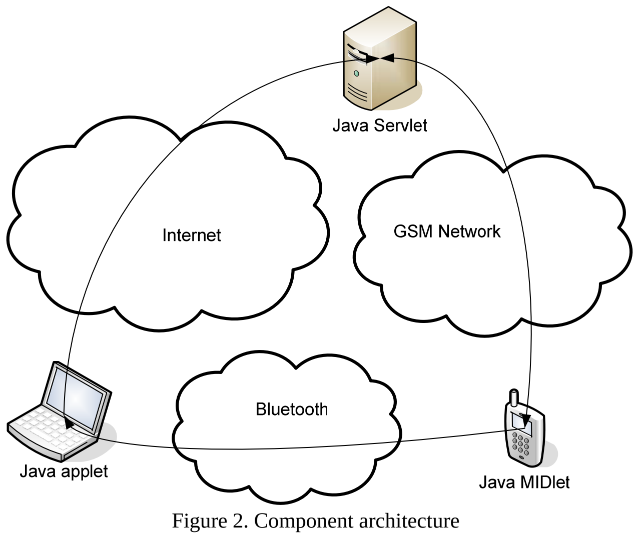


Figure 2. Component architecture

Fig. 2 illustrates the components architecture and communication interfaces used in the OTP solution. A Java MIDlet will be running on the mobile phone to handle OTP generation. A Java applet will run in the browser on the user's computer and relay the OTP from the MIDlet to the AS. The MIDlet and applet communicates over Bluetooth. A Java servlet will run in the AS and handle all client communication. The servlet will communicate with the MIDlet using short messages (SMS) over the GSM network and with the applet over a HTTPS connection.

A. *Java Midlet*

The java MIDlet is the core component on the client side. It enables the mobile phone to act as an OTP token and automatically exchange authentication messages with the AS and computer.

To be able to download the MIDlet to the mobile phone a user must register at the AS. This is done through an Internet page secured with TLS. When the registration is finished a push message is sent the user's phone which then automatically downloads and installs the MIDlet only requiring the user's agreement.

After installation the AS initializes a key exchange with the MIDlet by sending an SMS including some public values to the user's phone. Upon receiving this SMS the MIDlet is automatically started and the user is asked to enter two passwords.

The first password is chosen by the user and is used to ensure two-factor authentication, prevent misuse of stolen phones and act as an encryption key.

The second password is a short OTP generated by the server and displayed on the webpage when the MIDlet is downloaded. This is used once to authenticate the key exchange. The key exchange is further described later.

SMS communication

The Wireless Messaging API (WMA) [7] enables a MIDlet to automatically send and receive SMS messages. By using this together with the Push Registry function a MIDlet can be automatically activated upon receiving a SMS message on a specific port. The push registry's behavior can be described in three steps:

1. The MIDlet is registered with a port and a protocol such that, if any message arrives in the specified port with the protocol mentioned, the application management software (AMS) on the mobile phone delivers it to the MIDlet. In this case the protocol used is SMS over the GSM network. The registration is done statically using the Java ME application descriptor (JAD) file.
2. The server sends a message to the specific mobile phone using the particular protocol and port where the MIDlet application is registered to listen.
3. After the message is delivered to the mobile phone, the AMS calls the MIDlet application,

which has registered to listen to that particular port and particular protocol. Once the message is delivered to the MIDlet, it is the application's responsibility to process the message accordingly [8].

Key exchange

To make the OTP generation as secure as possible a strong secret key is used together with the challenge. This secret key must be shared between the client and server. The Simple Password Exponential Key Exchange (SPEKE) protocol [9] is used to securely exchange this key. SPEKE is an improved version of Diffie-Hellman (DH) [10] which prevents Man-In-The-Middle attacks by basing the DH generator on a hash of password. An attacker who is able to read and modify all messages between the client and server cannot learn the shared key and cannot make more than one guess for the password in each interaction with a party that knows it. A simple form of SPEKE requiring only two messages is illustrated in Fig. 3.

The AS starts the key exchange by generating a large and randomly selected safe prime p and computes

$$g = \text{hash}(s)^2 \bmod p$$

where s is the short OTP displayed in the browser after registration. Then the server computes

$$Ks = g^a \bmod p$$

where a is a secret random number and sends p and Ks to the MIDlet with an SMS.

Upon receiving the SMS from the server the MIDlet generates

$$g = \text{hash}(s)^2 \bmod p$$

$$Kc = g^b \bmod p$$

where s is the short OTP entered by the user at startup and b is a secret random number. Then the MIDlet sends Kc to the AS and computes the secret key

$$K = (Ks)^b \bmod p.$$

When the server receives Kc from the MIDlet it computes

$$K = (Kc)^a \bmod p$$

At this point the AS and the MIDlet shares a strong secret key K which can be used in OTP generation.

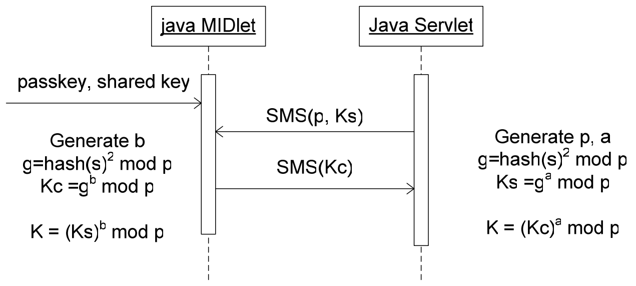


Figure 3. Key exchange

To prevent any attacker from learning the secret key it is encrypted with an expanded version of the user chosen password as an encryption key before it is stored in J2ME's Record Store[11]. The Record Store is used in private mode to deny any other MIDlet from getting access to the records. A hash of the user password is also stored so the MIDlet can verify that the correct password is entered at startup.

OTP generation

The OTP is generated from a hash of a concatenation of the challenge and the secret key.

OPT = hash(challenge || secret key)

When the user needs to be authenticated the AS generates a challenge and computes the corresponding OTP. Then it computes a message authentication code (MAC) of the OTP. The challenge and MAC is sent to the MIDlet as a SMS message while the OTP is kept at the AS.

As described in the previous part the MIDlet is automatically activated when the message arrives and the user is asked to enter his password. The MIDlet generates the hash of the password and checks it against the stored value. If they match the MIDlet uses the password to decrypt the secret key. The challenge and the secret key is concatenated and fed into the hash function to produce the OTP. Then a MAC is calculated over the OTP and checked against the MAC received. If the MAC does not match the authentication is aborted and the process must be restarted. If the values match the MIDlet starts a service discovery to set up a Bluetooth connection with the applet running on the computer. When the connection is set up the MIDlet sends the OTP to the applet. Then the connection is terminated and the MIDlet exits. Fig.4 illustrates the authentication process.

If a mobile phone without Bluetooth is used the user will have to enter the OTP in the browser manually. Then the OTP will be displayed on the mobile screen if the MAC is OK. Using a hash directly as an OTP is not suitable if the user must type it in manually. Therefore the hash must be truncated into an 8 byte long string and transformed into a more user-friendly format like a character string containing only letters and numbers.

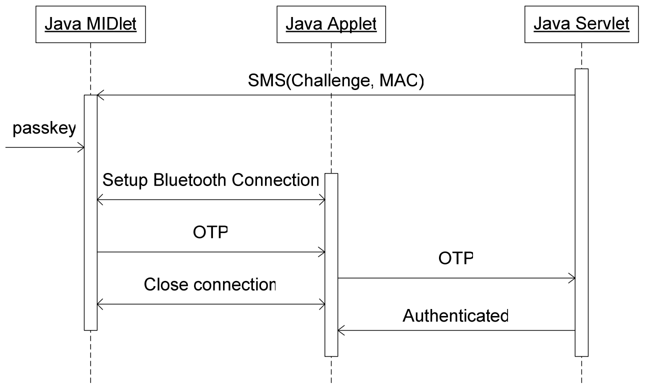


Figure 4. Authentication process

B. *Java Applet*

The java applet is necessary to enable automatic transfer of the OTP from the mobile phone to the computer. The applet will act as a Bluetooth server which the MIDlet can contact and setup a connection with. By realizing this component as an applet no additional software has to be installed on the user's computer. The mobile phone and computer must be paired before a connection can be setup. During the Bluetooth pairing a 16 byte pass phrase must be entered to secure the Bluetooth connection. When the applet has received the OTP from the MIDlet it contacts the servlet and sends the OTP. If the OTP is correct the applet is notified by the servlet and redirects the session back to the service provider.

C. *Java Servlet*

The Java servlet acts as the AS' interface towards the clients. When a user downloads a MIDlet the AS initiates the key exchange by sending its public values to the MIDlet by SMS. The servlet uses a SMS gateway to send and receive SMS messages. The gateway can be configured to send messages to specific ports. Therefore the servlet can send a SMS message to the port specified in the MIDlet. When the key exchange is finished the secret key is stored in the AS' database together with the user-profile. When authentication is requested the user is asked to enter his username. The AS gets the profile for this username and generates a challenge based on the profile. After generating the challenge the OTP and MAC is generated and the challenge and MAC is sent to the MIDlet by SMS. When the servlet receives the OTP from the applet it checks that it matches the OTP it created itself. If the values match the user is successfully authenticated. If the solution is part of a single sign-on environment an assertion is created that can be reused for several authentication procedures.

VII. Security Evaluation

This chapter contains a security evaluation of the solution. The evaluation is divided into four parts. Each part consists of the possible threats to this part and how the solution encounters these threats.

Fig. 5 illustrates the different components of the system that are vulnerable to attacks.

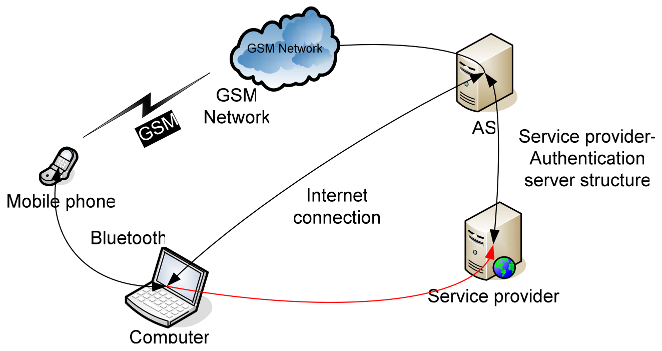


Figure 5. Threat model

A. *User devices*

- An attacker might try to steal a mobile phone to impersonate a valid user
- An attacker might try to tamper with a mobile phone to learn the secret key or password.
- Malicious software like worms and key loggers can gain access to sensitive information stored on the computer or typed on the keyboard. Malware is a big problem for everyone connected to the Internet.

Countermeasures

- The application is protected with a user chosen password. This results in a two-factor authentication preventing misuse of stolen phones.
- The secret key stored on the phone is encrypted with a strong encryption algorithm. A hash of the password is stored and not the password itself. The Java Record Store is used in private mode denying any other application access to the records.
- The user computer does not handle any secret information except when relaying the OTP. An OTP is protected against misuse by its properties.

B. *Bluetooth connection*

- An attacker might try to listen in on the Bluetooth connection to learn sensitive information

Countermeasures

- The Bluetooth connection is secured with the use of a passkey during paring. The only information being transferred over this connection is the OTP which is in itself secure.

C. Internet connection

- An attacker might try to impersonate a user by guessing the password.
- An attacker might try to hijacking the session to get unauthorized access.
- An attacker might try to eavesdrop on the connection to learn the password or other sensitive data.
- An attacker might try to mount a Man-in-the-middle (MITM) attack.

Countermeasures

- OTPs are not possible to guess since they appear as random numbers
- Eavesdropping, hijacking and MITM attacks are prevented by securing the connection to the AS with TLS

D. GSM Network

- A weakness in the GSM system is that that the server is not authenticated. This makes it possible to mount a MITM attack on the GSM network [12].

Countermeasures

- The key exchange is authenticated with a short OTP. This prevents a MITM for being able to learn anything useful during the key exchange.
- Tampering with the authentication values sent over the GSM network will only result in authentication failure since they are protected by a MAC. The MAC is also used to authenticate the server resulting in mutual authentication.
- Picking up the challenge and MAC sent over the GSM network will not enable an attacker to learn the key. The challenge is only a random number and the MAC is a hash generated of a strong cryptographic hash function which is not possible to revert.

VIII. Conclusion & Future works

This paper proposes a novel secure authentication solution based on the OTP principle. It shows how an OTP MIDlet turns the mobile phone into a secure and user-friendly authentication token. The proposed solution eliminates weaknesses of existing similar solutions such as the need for time synchronization, possibility of man-in-the-middle attacks and substantially improves the user-friendliness.

Using the mobile phone as an authentication token provides a very flexible and secure solution. It is therefore a very good alternative to the existing authentication solutions such as username and password.

The proposed solution can also be an alternative to existing OTP solutions which require extra hardware and therefore put an extra burden on the user and increase the costs for the companies/service providers.

This solution can be securely used in most services available on the Internet today which require authentication. This can be Internet banking, corporate intranet, Internet stores and e-Government applications.

Solutions that utilize the mobile phone for authentication are very likely to become a popular choice for many users and companies in the near future.

The paper also provides a thorough security evaluation which discusses the threat model and security properties of the proposed OTP solution.

As part of future works, the solution will now be properly integrated with an identity management platform and tested in a Single-Sign-On environment.

References

1. FreeAuthProject. The FreeAuth Project. [cited 2007 March]; Available from: http://www.freeauth.org/site.
2. NordicEdge. NordicEdge. [cited 2007 March]; Available from: http://www.nordicedge.se/.
3. Stallings, W., Network security essentials. 2003: Prentice Hall.
4. NIST, Electronic authentication guideline 2006.
5. M'Raihi, M.B., F. Hoornaert, D. Naccache, O. Ranen, RFC 4226 HOTP An HMAC-Based One-Time Password Algorithm. 2005.
6. Haller, N., et al., A One-Time Password System. 1998, IETF.
7. Ortiz, C.E. The Wireless Messaging API. 2002 [cited 2007 March]; Available from: http://developers.sun.com/techtopics/mobility/midp/articles/wma/.
8. Roy, S. Push messages that automatically launch a Java mobile application. 2006 April 17th [cited 2007 March].
9. Jablon, D., Strong Password-Only Authenticated Key Exchange. Computer Communication Review, ACM SIGCOMM, 1996. vol. 26(no. 5).
10. Rescorla, E., RFC 2631 Diffie-Hellman Key Agreement Method. 1999.
11. Microsystems, S. Java ME Technology APIs & Docs. [cited 2007 March]; Available from: http://java.sun.com/javame/reference/apis.jsp.
12. U.Meyer and S.Wetzel, A Man-in-the-Middle Attack on UMTS. In Proceedings of the 2004 ACM Workshop on Wireless Security, 2004.