HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

Pranav Kumar Sharma

Short-Lived Certificates as a Mobile Authentication Method

Master's Thesis
Espoo, 26 June, 2009

Supervisors: Professor Antti Ylä-Jääski, Helsinki University of Technology
Professor Christian Probst, The Technical University of Denmark
Instructor: Henri Mikkonen, M.Sc. (Tech.)

HELSINKI UNIVERSITY
OF TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Programme of Security and Mobile Computing

ABSTRACT OF THE
MASTER'S THESIS

Author:	Pranav Kumar Sharma	
Name of the thesis:	Short-Lived Certificates as a Mobile Authentication Method	
Date:	26 June, 2009	**Number of pages:** 69
Professorship:	Data Communications Software	**Code:** T-110
Supervisors:	Professor Antti Ylä-Jääski Associate Professor Christian Probst	
Instructor:	Henri Mikkonen, M.Sc. (Tech.)	

Convergence of mobile phones and the Internet opens up unlimited opportunities to service providers and users alike. For service providers, it offers an opportunity to deploy innovative services for a large base of users and boost revenue. For users, it brings them closer to the dream of "anytime-anywhere"services available in "all-in-one"device. However, exploiting these opportunities into real world requires an assurance to both service providers and mobile users that the services adhere to well known security models.

The Public Key Infrastructure for mobile phones (mobile PKI) is emerging as enabling technology for accessing services over the Internet. Yet, authentication of service providers and users in the mobile PKI brings its own set of challenges. For example, check for certificate revocation is vital but it is resource consuming and requires connectivity. Security of the private key is central to PKI but portable nature of mobile phones exacerbate the issue. Existing practices of offline credential verification to issue a certificate restrict usability of certificates for mobile users.

This thesis proposes an authentication method based on short-lived standard X.509 certificates. The objectives are to minimize need for certificate revocation checks, secure credential store and online certificate request, verification and issuance. We present an architecture for obtaining certificates online to mobile phones using existing credentials. We also provide a mechanism for secure storage and usage of credentials in mobile phones so that the mobile signatures qualify for European Telecommunication Standards.

We focus on specific usage scenarios and lay down the system requirements in terms of usability, technical merits and use of open standards. After the description of the solution, we present a prototype implementation and illustrate how the usage scenarios benefit from the proposed method. We analyze and validate the solution against requirement criteria using the Goal Matrix Approach.

Keywords: PKI, authentication, X.509 certificates, mobile signatures, smart card, SATSA

Acknowledgements

I would like to acknowledge the Helsinki Institute of Physics Technology Program at the European Organization for Nuclear Research (CERN) for supporting my thesis work in every possible way.

Several people have contributed to the thesis. I would like to thank Henri Mikkonen for instruction me through the thesis. I would also like to thank Professor Antti Ylä-Jääski and Professor Christian Probst for supervising my thesis.

I have received valuable feedback and hints during the writing and implementation processes of the thesis from my colleagues Tapio Niemi, Kalle Happonen and John White. They also deserve my thanks.

I would like to dedicate this thesis to my parents whose support made the studying possible.

CERN, Geneva, Switzerland, 26 June, 2009

Pranav Kumar Sharma

Table of Contents

List of Figures

List of Tables

Abbreviations

3GPP	3rd Generation Partnership Project
AP	Application Provider
APDU	Application Protocol Data Unit
API	Application Programming Interface
CA	Certificate Authority
CBC	Cipher Block Chaining
CDC	Connected Device Configuration
CIA	Confidentiality Integrity Availability
CLDC	Connected Limited Device Configuration
CLIP	Customized Lightweight Identity Provider
CM	Credential Manager
CMS	Cryptographic Message Syntax
CP	Client Proxy
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DES	Digital Encryption Standard
DN	Distinguished Name
DSA	Digital Signature Algorithm,
ECB	Electronic Code Book
EGEE	Enabling Grid for E-sciencEs
ETSI	European Telecommunication Standards Institute

GAA	Generic Authentication Architecture
GQM	Goal Quality Matrix
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IP	Internet Protocol
IdP	Identity Provider
JSR	Java Specification Request
MIDP	Mobile Information Device Profile
MSS	Mobile Signature Service
MSSP	Mobile Signature Service Provider
OCSP	Online Certificate Status Protocol
OS	Operating System
OTA	Over The Air
PC	Personal Computer
PDA	Personal Digital Assistant
PEM	Privacy Enhanced Mail
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
RA	Registration Authority
RAM	Random Access Memory
RFC	Request For Comment
ROM	Read Only Memory
SAML	Security Assertion Markup Language
SAT	SIM Access Toolkit
SATSA	Security and Trust Services API
SCLS	Short Lived Credential Service
SE	Security Element
SIM	Subscriber Identity Module
SLCS	Short Lived Credential Service
SMS	Short Message Service

SP	Service Provider
SSCD	Secure Signature Creation Device
TPM	Trusted Platform Module
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USAT	Universal SIM Access Toolkit
WAP	Wireless Application Protocol
WIM	Wireless Identity Module
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WTLS	Wireless Transport Layer Security
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

Chapter 1

Introduction

Mobile phones are increasingly being used for accessing a variety of services over the Internet. These services include e-commerce applications such as mobile-banking, remote management of personalized data such as remote log-in and email services and device-device message exchanges among others. The authentication of involved entities (mobile users and service providers) and confidentiality of data exchange are vital for practical usage of such services [31]. At the same time, Public Key Infrastructure (PKI), based on the concept of the trusted third party has emerged as the principle security mechanism for services and applications over the Internet [22].

Seamless access to Internet services with mobile phones requires conformance to established security practices in Internet. Indeed, PKI as enabling technology for mobile devices has been advocated by Dankers et *al* [18] among others [35, 31].

Adapting PKI for mobile devices (termed as mobile PKI) poses several unique challenges [31]. Relatively limited computing and storage resources of mobile devices and restricted bandwidth make it difficult to implement traditionally intensive PKI services such as key-pair generation, certificate maintenance and to deal with Certificate Revocation Lists (CRL). Processing CRLs in mobile devices only adds more resource requirements to the constrained environment of mobile PKI. Even as alternatives of CRLs such as Online Certificate Status Protocol (OCSP) [38] are available, it does not eliminate the need to check the certificates against revocation and requires connectivity to OCSP Server during authentication process [27].

Protecting private key is critical in any PKI based security mechanism. A compromise of private key can defeat most elegant authentication solutions [33]. As mobile devices are easily accessible and vulnerable to misuse, securing private keys in mobile devices are even more crucial for any PKI based authentication.

Another challenge is online (and real time) request and issuance of certificates for mobile devices. Even as standards for certificate request, e.g., Public Key Cryptography Standard #10 (PKCS#10) [5] and response, e.g., PKCS#7 [6] exist and can be performed online, the process of credential verification continues to be offline and organization specific [52]. This practice is time consuming, costly and restricts the usage of PKI, in particular for mobile device users.

Short-lived certificates [17] may address some of the aforementioned challenges to mobile PKI. These certificates are standard X.509 certificates [49] except that validity period is less than 1 million seconds (approx. 11 days). The short-lived certificates can effectively address the challenge of CRLs. For instance, the service provider (or any entity requesting authentication) may not require to check the certificate against revocation owing to the short-lived duration of certificate. Even though the short-lived certificates do not completely eliminate the possibility of certificate being compromised, they do restrict the time for misuse and they do ease the revocation requirements.

Short-lived certificates are used to authenticate some EGEE [1] Grid users and provide strong motivation for online certificate request and issuance. The certificate request and issuance by Grid Middleware is executed in single session using user's home Identity Provider (IdP) which already possess the user's credential. This mechanism can be equally applied to real time request and issuance of certificates to mobile device users. Indeed, if the mobile user's home IdP (for instance mobile operator, user's national IdP) is trusted by CA issuing short-lived certificates, this is entirely feasible.

Short validity period and secure credential store combined with mechanisms for real time request and issuance of short-lived certificates generates novel usage scenarios for mobile authentication. As we demonstrate in subsequent Chapters; disconnected scenarios and mobile travelers benefit significantly from our proposed solution. Moreover, the secure credential store features of our solution qualifies for European Telecommunication Standards Institute (ETSI) [20] framework for mobile signatures.

The thesis is organized as follows. We explain the problem statement of the thesis in Chapter 2. A literature review in the context of the problem is discussed in Chapter 3. The requirements of the thesis and criteria to evaluate the solution are established in Chapter 4. We explain our solution in Chapter 5 and provide a prototype implementation in Chapter 6. We evaluate the solution against the requirements (and criteria) in Chapter 7. Finally, we conclude the thesis and suggest pointers for future work in Chapter 8.

[1] Enabling Grid for E-SciencEs (EGEE) is an European Union initiative to develop Grid infrastructure for scientists. See http://www.eu-egee.org/.

Chapter 2

Problem Statement

2.1 Definition of Key Terms

This Section discusses key terms which will feature in subsequent Sections of the Chapter.

Authentication: The term "authentication" refers to the process of establishing identity of an entity (e.g., a person, a message, a system) conclusively. Oxford Dictionary [1] defines the term as "To prove that something is real, true, or genuine". Searchsecurity.com, a website [2] for security information, defines authentication as "A process of determining whether someone or something is, in fact, who or what it is declared to be".

X.509 Certificate: This is a format of digital certificate defined by ITU-T[3] and the standards are detailed in RFC 5280 [4]. The central charactertiscs of these digital certificates are hirerachical structure which follows from the X.500 format.

Short-lived certificate: EGEE Grid Middleware defines short-lived certificate as "Standard X.509 certificates with a lifetime of less than 1 million second (approximately 11 days)" [17]. The term short-lived certificate is also generally used in literature [25] to refer to a certificate with "short life time" with no clear definition in terms of certificate

[1] A publisher of English dictionaries. See http://www.oup.com/elt/catalogue/teachersites/oald7/?cc=global

[2] See http://searchsecurity.techtarget.com/sDefinition/

[3] ITU-T stands for Telecommunication Standardization Sector. ITU-T defines telecommunication standards on behalf of International Telecommunication Union (ITU)

lifetime or the format of the certificate. However, in this thesis, we follow the definition by EGEE when referring to short-lived certificate.

Certificate Authority (CA): A trusted third party, that issues and manages digital certificates [21]. The issuance means that the digital certificate is signed by the CA. The management of certificates include re-issue of certificates, e.g., after expiry of validity period, and revocation (which may be requested by client). CA requires credential verification of requester before issuing a certificate and this function is usually performed by Registration Authority (RA).

2.2 Research Problem

The goal of this thesis is to research short-lived certificates as mobile authentication method. The motivation for the research stems from usage of short-lived certificates in authenticating researchers and scientists for Grid access in several countries. An important feature in the usage is that prospective Grid users are registered with a home IdP and they do not have to register with the Grid Security domain. Besides, the users can obtain certificates online.

We believe that short-lived certificates in mobile phone can benefit novel usage scenarios. The target of the research is twofold:

1. Prototype interaction between CA and mobile device to obtain short-lived certificate.
2. Mechanism for secure storage of private key at mobile node.

2.3 Research Method

Approach to a research problem can range from descriptive characterization of the problem (qualitative approach) to abstract modelling of the problem (quantitative approach). Qualitative approach is widely used for exploratory research, for example to understand issues in social sciences and such issues often have multiple perspectives [30]. A qualitative researcher organizes the observations about the problem, create generalizations and defend them through arguments and reasoning from the real world. On the other hand, quantitative approach requires defining and constructing an abstract mathematical model to represent the problem. The solution is sought through formal proofs with rigorous mathematical treatment (e.g., statistical techniques) of the model.

Constructive Research as a research methodology is an established practice in software systems that incorporates aspects of qualitative and quantitative research. The approach can be summarized as follows (See [48] for details):

- Identify a practically relevant problem with a research
- Obtain a comprehensive understanding of the problem
- Construct a solution idea and demonstrate that solution works
- Show original research contribution and theoretical relevance
- Examine the scope for practical usage

The characteristics of Constructive Research motivate us to select this methodology for the research problem. Additionally, well defined phases serve as useful guidelines for the research. In line with the Constructive Research methodology, we formulate following phases to approach the research problem:

1. *Literature review:* We study existing literature on mobile authentication methods to identify challenges involved as well as to find out relative strength and weaknesses of different approaches. The proposed authentication method builds on existing literature base and attempts to determine what benefits it could offer in comparison to the existing ones.
2. *Solution Construction:* We identify the requirements that follow from initial motivation (as outlined in Chapter 1) and subsequent literature review. For the construction, we draw lessons from established software design approaches [14, 42] ensuring that the solution confirms to the requirements.
3. *Prototype Implementation:* We implement a solution prototype to demonstrate the work to end users in a practical setup and to describe different use case scenarios. Also, the prototype serves to illustrate low level details of the proposed authentication method.
4. *Analysis and Evaluation:* We analyse the solution against the requirement criteria. We plan to be as objective as possible in our analysis and draw lessons from the established software evaluation approaches [42].

2.4 Scope of Thesis Work

We describe the scope of the thesis in context of the thesis objectives. The objectives are as follows:

A To demonstrate the benefits of short-lived certificates using novel usage scenarios. For example, minimizing the needs of revocation checks.

B A mechanism for obtaining short-lived certificates in mobile device online over the Internet. Obtaining a short validity certificate is useful if it can be obtained online and in real time.

C A solution for secure storage of private keys in the mobile phone.

To fulfill objective A, we focus on selected usage scenarios and illustrate the authentication method through use cases and sequence diagrams.

To address objective B, we assume that mobile users do have an existing IdP that can facilitate online credential transfer for real time issuance. We limit the problem in terms of interaction between 3 entities: a mobile user, an existing IdP to which the user is registered and the CA. The entities communicate over the Internet to facilitate the objective B.

For objective C, we select an existing storage technology through an extensive literature review and motivate our selection from the system requirements. The thesis, however, details the interfacing between storage component and the the proposed system.

Chapter 3

Literature Review

This Chapter provides an overview of mobile authentication. We begin with identifying key issues in mobile authentication, discuss salient features of different methods, their goals and suitability to specific contexts as well as comparative advantages they offer. We also discuss present challenges and future directions.

3.1 Mobile Authentication

Mobile authentication has been discussed widely and rich literature exists on solutions based on symmetric key, asymmetric key or a combination of these two [11, 36, 26]. Before proceeding further, we briefly discuss symmetric and asymmetric cryptography.

Symmetric Key Cryptography The underlying fact in symmetric key cryptography is that encryption key can be derived from decryption key and vice-versa [47]. If encryption key and decryption key in symmetric cryptography are same (which is generally the case), it is termed as secret key cryptography.

Asymmetric Key Cryptography Also called public key cryptography, each party has a public key and a private key. Public key, as the name suggests, is public and known to others. Pair of public and private keys are complementary. A message encrypted by public key can only be decrypted by associated private key and vice-verse. Which means if Alice wants to send a message to Bob, she encrypts the message using Bob's public key. Assuming Bob's private key is known only to Bob, only Bob can decrypt the message sent from Alice. However, in practice, encryption by the

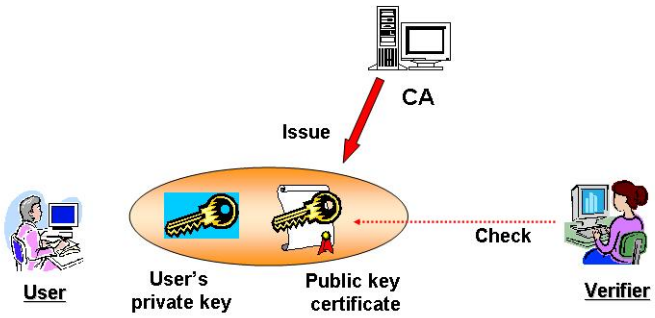

Figure 3.1: Basic Elements of PKI [1]

private key is used for signing messages (for message integrity) and proof-of-possession of private key.

Public Key Infrastructure Administration and management of shared keys such as generation, distribution and storage becomes complicated with increase in participating entities [18]. Public Key Cryptography addresses these challenges by reducing the number of keys required by each party (instead of 2**n only n key pairs are required, n being the number of participating entities). Further, the concept of public keys make it possible to put all public keys at a dedicated place accessible to all. Of course, public key mechanism brings challenges in terms of who should authorize, maintain and associate a public keys to participating entities. Public Key Infrastructure (PKI) has been developed precisely to address these challenges. A central component of PKI (see figure 3.1) is the CA which issues public key and binds the key to an entity. This activity is more formally known as issuing *digital certificate*. A digital certificate contains more than just public key (see figure 3.2). PKI is a natural choice for security mechanism in Internet with continuous evolving services and complexity.

.

Authentication requirements in mobile communications have been identified as early as since the evolution of mobile systems itself [44]. In the beginning, as mobile usage was limited to voice communications and messaging only, secret key arrangement between mobile subscriber and home operator was principle security mechanism. This was achieved with pre-established secret key installed on Subscriber Identity Module (SIM)

[1]The image is referred from IT and Security Laboratories at Kyushu University, Japan. See http://itslab.csce.kyushu-u.ac.jp/sakurailab/research/pki.html

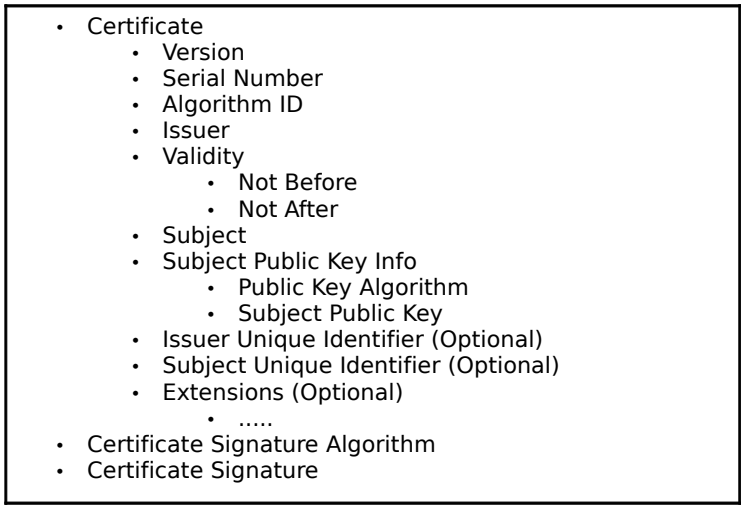

Figure 3.2: Structure of the X.509 v3 Certificate

issued by home operator during subscriber registration. Authentication was achieved using challenge-response messages which fundamentally test the other party's ability to encrypt or decrypt (response) a given message (challenge).

However, the attempts to address key security principles, normally described in terms of Confidentiality, Integrity and Availability (CIA) model, have become increasingly challenging with evolving usage of mobile devices for accessing services over the Internet. The early security solutions based on pre-established keys between operator and subscribers are infeasible in present scenario which involve many more entities than just mobile subscriber and home operator. PKI as enabling technology for present and future mobile systems have been advocated in [31, 18, 10] among others. Indeed, existing literature provides a clear indication for PKI based security solutions for several reasons:

1. PKI is already an established security mechanism for services and transactions in Internet [10, 18, 35].

2. For accessing services over Internet, mobile users need to comply with existing and established security mechanisms (digital certificates, digital signatures) [46].

3. It is not practical for mobile users to possess pre-established secret keys for growing number of service providers in Internet [18, 34].

One of the motivations of the thesis is to provide authentication (of mobile users) for services over Internet. Moreover, the work uses existing security features and applications for Internet such as HTTPS/TLS as underlying security framework for the

proposed authentication. We therefore focus on review of PKI in mobile devices.

3.2 PKI in Mobile Devices

Adapting PKI for mobile sets brings specific challenges mostly owing to limited resources in mobile devices, less bandwidth availability and heavy certificate validation process (common for wired networks) [35, 18, 31]. For instance, mobile devices must have capabilities to generate a key pair, compute and verify digital signatures, compute certificate path and store certificates as well as revocation lists [4]. This requires significant processing and storage capabilities in context of mobile phones. PKI functions in mobile devices can be summarized as below:

1. Generation of key-pair (public and private keys)
2. Sending Certificate Signing Request (CSR) to CA
3. Receiving and storing certificate issued by CA
4. Validating 3rd party certificates
5. Digital signature generation and verification
6. Functions for encryption and decryption

Improved computing and storage resources in mobile devices have made it convenient to generate key pair. However, online CSR is a major challenge for the fact that it requires some form of offline credential information of mobile user with CA [39]. Several bootstrapping specifications have been proposed for requesting and receiving online identity (e.g. certificates) for mobile users.

In wired networks, certificate validation requires accessing and examining certificate chain and Certificate Revocation List (CRL). The validation is considered successful if all the certificates in the certificate path (from the leaf to the root of tree) are checked and ensured that none of them have been revoked. This process is heavy on resources and time consuming and is not very suitable for mobile devices. Several mechanisms have been suggested to minimize the validation process or entirely eliminate the need of validation [35] for mobile clients. Some examples are:

A. Dedicated servers to verify the CRL

Online Certificate Status Protocol (OCSP) [38] provides an alternative to CRL. Dedicated OCSP servers can accept requests for certificate validations and respond if the

certificate is valid or not, taking away the burden of obtaining and processing CRLs from mobile clients.

B. New certificate specifications

Doing away with the need of certificate verification completely has been implemented in Wireless Transport Layer Security (WTLS) for Wireless Application Protocol (WAP) server authentication. As mobile clients using WAP applications might require server authentication, WAP servers are issued with short-lived certificate by CA for short duration (e.g., for 24 hours). WAP servers use this certificate to authenticate themselves to mobile clients. As the server is short-lived, mobile clients do not have to implement either CRL or OCSP for WAP server authentication.

All the major mobile platforms such as Symbian and Windows Mobile offer PKI functionality namely key pair generation, CSR, credential store, signature generation and verification as well as cryptographic library. Symbian's Certificate Manager, along with aforementioned functions provides flexibility to store certificates, private keys and other credentials either in device or in temper resistant smart cards as well as facility to import and delete keys. Java Specification Request (JSR) 177, also named as Security and Trust Services (SATSA) API for Java Mobile Edition (Java ME), offers PKI features similar to Certificate Manager. SATSA provides flexible "Security Element" concept to support credential store and cryptographic operations in SIM and external smart cards. Both Certificate Manager and SATSA support several cryptographic algorithms (e.g. RSA, DSA, Diffie-Hellman) and hash functions for CSR and digital signatures. Both SATSA and Symbian Certificate Manager provides native API for crypto functions as well as support external cryptographic libraries for PKI functions in mobile devices.

3.3 Securing Private Keys

Importance of securing private keys is crucial for success of any PKI technology [10, 44, 46]. A survey of contemporary methods for secure storage of keys in mobile devices and their comparative advantages are discussed in [46]. We discuss the major solutions in following Subsections.

3.3.1 SIM Based Technologies

SIM Application Toolkit (SAT) is an existing technology for developing device and card independent application [50]. These applications are allocated on SIM card and

are capable of interacting with device set. An enhanced version of SAT technology is Universal SAT (USAT) which is targeted for third generation mobile phones and allows the possibility of opening up Hyper Text Transfer Protocol (HTTP) connections to communicate over IP Internet.

However, several disadvantages have been cited which restrict practical usage of this technology [46]. The applications are deployed as applets and need to be downloaded "over the air". This is a time consuming process because data in mobile devices can be downloaded only in short chunk of messages (short messages). The applet therefore needs to be fragmented into several linked short messages. Further, the installation of applets is sensitive task in itself involving security risks and performance issues [45].

Wireless Identity Module (WIM) is another SIM based solution which ensures that key pairs are generated inside the card and private keys never go outside the card [51]. However, this technology requires that WIM application is included in the SIM by the manufacturer.

3.3.2 Handheld Based Technologies

Handheld based solutions for key generation and storage are more diverse than SIM based solutions [46]. Notable among these solutions are Symbian Operating System (Symbian OS), Windows Mobile OS and Java ME

Symbian OS provides security of key generation and storage with two security components namely Certificate Manager and Cryptographic Module [23]. Certificate Manager offers extensive Application Programming Interface (API) for key generation, storage and retrieval of certificates. This component also facilitates assigning trust level to a certificate for individual applications. The Cryptographic Module provides a range of algorithms for hash functions, symmetric and asymmetric ciphers and random number generations which can be used with Certificate Manager. It must be noted that this solution is available only for handsets which are Symbian OS based.

Windows Mobile OS [2] key generation and storage in this platform are offered through Crypto API and Cryptographic Service Providers (CSP). Crypto API interacts with OS which in turn communicates with CSPs. CSP is an independent module which offers cryptographic implementations of algorithms for key generation, storage and verification. The use of external Crypto API is also facilitated. Moreover, the OS

[2] For a comparison of security features of Mobile Phone Operating Systems, see [32].

provides facility to integrate with external smart card with smart card CSPs. These smart cards can be used for storing private keys [18].

Java ME Being platform independent, Java ME facilitates application development on a wide variety of hand-held devices. Development on different platforms is supported through configurations and profiles. A configuration is essentially a specification consisting of a virtual Machine and a set of standard Java APIs to allow application development on a set of devices. Two main configurations of Java ME are Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). As the name suggests, CLDC targets devices with limited resources such as mobile sets and low end PDA. CDC is designed for more resource capable devices such as high end PDAs and embedded devices. A profile is another set of APIs on top of configuration and targets specific device. Mobile Information Device Profile (MIDP) is a profile for mobile devices used to develop user applications. A combination of configuration and profile thus ensures a portable application development referred to as "MIDlet" on a range of hand-held devices.
Security and cryptographic capabilities on MIDlets are provided through a variety of Crypto APIs [9]. Bouncy Castle and IAKI Micro Edition are some of the commonly used APIs for this purpose. Section 3.3.4 describes a new set of APIs in Java ME for secure communication with SIM and smart cards.

3.3.3 Hardware Assisted Secure Environment

This approach uses dedicated hardware with on chip memory and processing for secure and isolated execution of trusted applications. A secure execution environment has following characteristics:

- Code execution takes place in an isolated environment different from normal execution environment in the same device.
- The hardware contains a dedicated on-chip memory to facilitate trusted execution and to avoid any attack that is possible owning to common memory.
- The environment possesses secure credential store for persistent data (such as private keys for the certificates).
- Finally, only trusted applications have the access of environment (including data and computation) so that integrity of secure environment is not compromised. that

Some of the commercially available (and commonly used) secure environments are following:

M-Shield provided by Texas Instruments implements secure environment by "secure processor mode" and contains features such as on chip RAM and ROM (see [8] for more on M-shield). Further, the execution in processor mode is isolated from OS and other applications running in the device.

Trusted Platform Module (TPM) allows execution of pre-defined code in secure environment [15]. TPM are essentially hardware modules with own processor. TPMs are most commonly used for "platform authentication". However, TPMs (at least commercially available at present) use device OS to provide secure environment and thus OS is part of trusted code base. Also, TPMs are vulnerable to memory attack as the TPMs shares device memory for execution.

Nokia On-borad Credentials are set of specifications developed by Nokia which uses *already existing* secure hardware environments (such as M-Shield and TPM) to develop an architecture of secure storage and usage of credentials [19]. The designers of specifications have argued that On-board credentials enable the developers to deploy credential algorithms without seeking approval from secure hardware manufacturer. At the same time, it also prevents secure platform from executing malicious credentials algorithms that might steal credentials from device (see [19]). The On-board credentials thus allow secure chips to hold credentials from "multiple sources".

3.3.4 Hybrid Solutions

SIM based solutions are characterized by high security but with low processing power. Handheld based solutions take advantage of rich user interface and processing power but offers comparative lower level of security. Hybrid solutions attempt to combine advantages of these two extremes.

WML/XHTML Script Wireless Markup Language (WML) [24] and eXtensible Hyper Text Markup Language(XHTML) [28] are languages for web application development on mobile sets. These languages contain cryptographic libraries that can be executed on mobile browser and can establish session with WIM application residing inside smart cards. For instance, these libraries contain a function named "Sign Text"

which supplies plain text to the WIM application. WIM returns the signed text after applying crypto functions with private key residing in the smart card.

However, a major restriction with this solution is that the manufacturer has to provide functionality for communication between WIM and the script running in browser.

SATSA is an emerging solution based on Java ME [40]. Essentially, SATSA provides a communication channel between a Java ME application such as MIDlet running on top of mobile OS and applications running inside SIM (e.g. WIM application). The electronic signature is computed by applications running inside SIM card which ensures that private keys never leave the SIM card. There are 3 different communication modes between Java ME applications and SIM that are supported by SATSA depending on type of SIM application. This solution is poised to evolve across different mobile set vendors as it is device independent (requires Java ME) and can communicate with a variety of SIM applications. See [29] for SATSA specifications.

In table 3.1 we compare SIM based, Handheld based and hybrid solutions on factors such as ease of usage, need for manufacturer support and level of security offered.

3.4 Certificate Request, Issuance and Identity

Requesting and obtaining a certificate is crucial step in a PKI based authentication. Although standard procedures for certificate request to CA (e.g. PKCS#10 format) as well as response from CA (e.g. PKCS#7 format) has been developed, the verification of request (and associated credentials) lacks standards [12]. The verification is usually performed off-line and depends on organization's policy to which CA belongs.

In rest of the Section, we describe the existing mechanisms of trust sharing and bootstrapping of credentials and how they make the process of certificate request and issuance real time, scalable and convenient for users.

3.4.1 Online CA

PKI based approach for providing services in organizations are traditionally limited because different services (authorizing for e-mail service, remote login to specific servers) require separate certificates and the process of request and issuance of certificates are mostly offline thus costly and time consuming. The term Online CA, means that certificate request and issuance can be online *without* requiring offline verification [21].

Table 3.1: Comparision of Available Solutions

Solution Class	Technology	Security Features	Limitations
SIM Based	SAT	secure execution environment	applet download via OTA Server, slow and error-prone.
	WIM	keys generation inside the card, private key never leaves the card.	Requires manufacturer support.
Handheld Based	Device OS (e.g. Symbian, Windows OS)	key is stored in file system managed by OS, key security through crypto API.	no interoperability across OS.
	Java ME	provides extensive crypto API (e.g Bouncy Castle)	Need support from underlying OS.
	Hardware assisted secure env.	isolated execution often using on-chip memory and processor	Cost, No interoperability
Hybrid	WML/XHTML	Communicates with WIM application for signing text	Requires browser support from manufacturer to interact with WIM.
	SATSA	Extends Java ME with new API to communicate with SIM and smart cards for key generation and store, inherits all features available in Java ME.	Manufacturers need to comply with APDU commands.

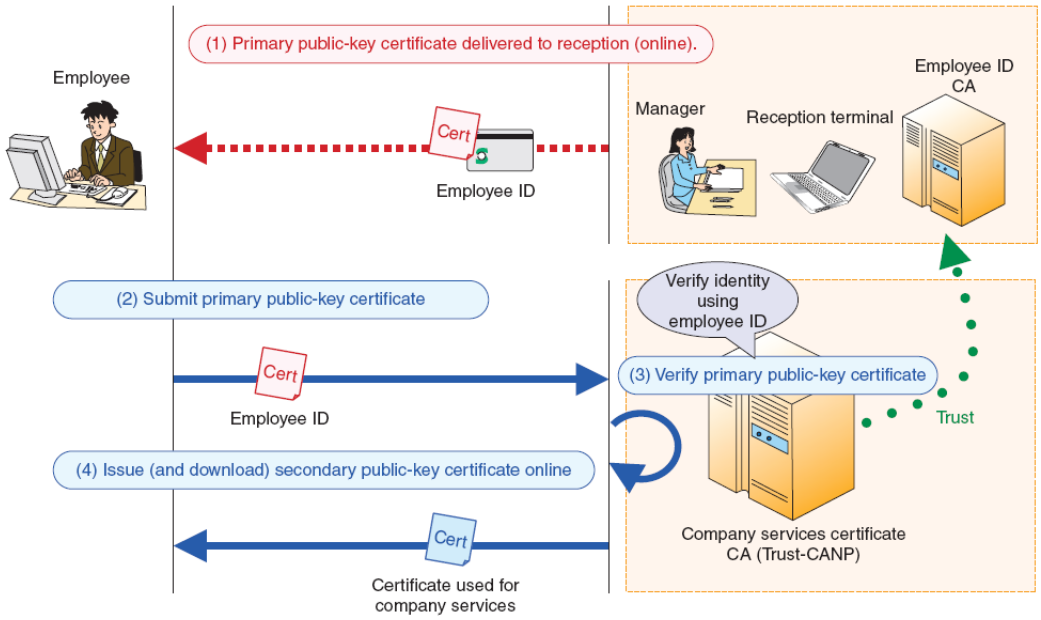

Figure 3.3: Online Certificate Issuance in NTT Inc., Japan [39]

As we will see shortly, in certain conditions, online CA is feasible and offers several benefits.

An organization possesses credentials of its employee (or users) in some form obtained during registration (verified offline for example against some official ID such as Driving licence). If the credential information is stored in a server (credential server) and made accessible online to RA responsible for verifying certificates, offline credential verification can be eliminated. A certificate service can be made available to users which accepts certificate request and interfaces with RA and credential server

A customized Lightweight Identity Provider (CLIP) by Nakajima et.*al.* [39] exploits credential server feature in NTT, Japan, to offer online certificate to its employees. A CLIP wrap server serves as interface for users requiring online certificates and is connected to credential server. The credential server in turn is connected to several issuing authorities to issue certificates pertaining to different services online. The organization implements several auxiliary services to employees based on certificates (see Figure 3.3.)

This approach offers several benefits to both users and the organization. A flexible format and variety of user data independent to any CA's requirement can be stored in central credential system. Several certificate formats can be supported as credential system is linked to many Issuing Authorities. Moreover, flexibility in format and

credentials ensure that a certificate containing only required information is issued.

Also, the CA services can be outsourced to third party keeping only central credential system in organization's control. Thus CA can act as application service provider. Moreover, if an online interface is provided for third party operators, this could enhance the online CA services to end users of organization.

Online Certificates in EGEE Grids

gLite, an EGEE Grid Middleware, provides online Short-lived certificate request and issuance through a service known as Short-lived Credential Services (SLCS) [17]. This service is available to Grid users who already have credentials with their home IdP (for instance a university log-in system) provided home IdP maintains a trust relation with Grid Middleware. The certificate then can be used by users to authenticate for Grid services. The process is briefly described below:

1. Prospective Grid user sends a login request to SLCS.
2. SLCS in turn asks for authentication entity (Federated IdP) of the user.
3. User selects the Home IdP and is redirected to that IdP for actual authentication. The redirection returns a handle back to SLCS.
4. The IdP sends the security assertions (transparently to client) to SLCS.
5. SCLS generates an authorization token and sends back to user.
6. User sends a CSR message along with authorization token to SLCS
7. SLCS in turn verifies message (signature and authentication token) and forwards it to online CA for certificate issuance.
8. Online CA issues the short-lived certificate for client which is forwarded by SLCS from CA to client.

A significant advantage for users is that the process is real time and they do not have to prove their credentials separately to Grid authorities. For Grid authorities, it is cost effective as they do not have to deploy credential verification systems for potentially large number of researchers.

3.4.2 Federated Identity Providers

Sharing of trust among different security domains reduces the cost and time in request verification. More importantly, trust sharing eases the burden from users to maintain

and provide several credentials facilitating Single Sign-on. Trust sharing is highly scalable and is an established practice in current PKI implementations [43].

Federated Identity Management attempts to use single identity information across multiple security domains. In addition to security and usability, the issues of interoperability, privacy, liability and access policy are primary challenges in any Federated Identity Management. Liberty Identity Federation by Liberty alliance [3] (of more than 30 organizations) is one such initiative to bring the federated identity management into real world.

Shibboleth is yet another Federated Identity Management solution developed by Internet2 Middleware [37]. The solution is provided in form of a specification and open source implementation. It is basically an HTTP POST based Single Sign-On service.

3.4.3 Bootstrapping

Cellular networks are already used to authenticate millions of subscribers worldwide. However the same infrastructure has not been exploited fully to extend the authentication as a service to third part service providers. A lack of well defined standards is one of the reasons among others [34]. Generic Authentication Architecture (GAA) as part of Release 6 specification of 3rd Generation Partnership Project (3GPP), enables cellular operators to extend the mobile user authentication to access new services [7]. Such services may be owned by cellular operators or by third party service providers who trust the cellular operators. GAA framework allows extension of existing authentication mechanism for these new services.

Two main activities (see Figure 3.4) in GAA are:

- Bootstrapping of credentials from cellular authentication system
- Use of the bootstrapped credentials (GAA credentials) to authenticate to third party.

Bootstrapping procedure can be initiated by either application server or mobile user. GAA specifies a new bootstrap server to facilitate bootstrapping of credentials. If the bootstrapping procedure is successful, GAA credentials (GAA master session key, transaction identifier) are stored in SIM as well as in authentication center of cellular operator. A service provider accesses the mobile user's credentials stored in authentication center (using transaction-id) to verify the credentials supplied by mobile user.

[3] See the homepage at http://www.projectliberty.org/

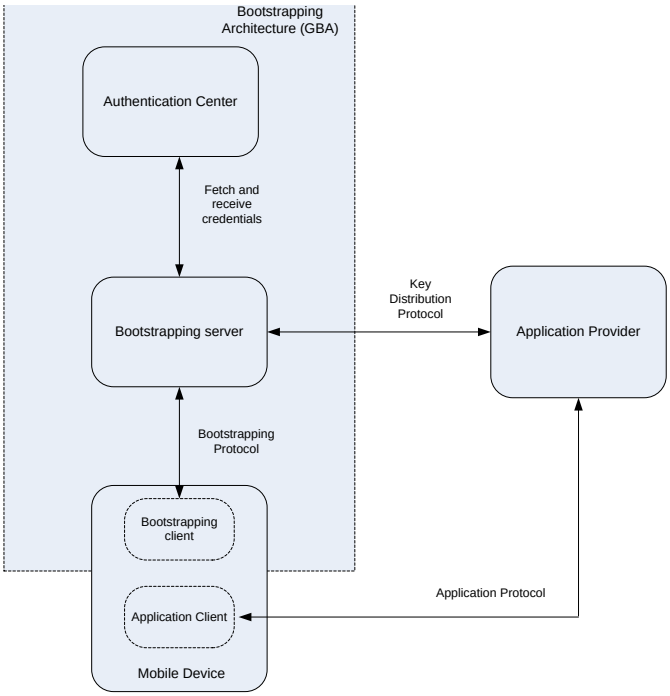

Figure 3.4: Generic Authentication Framework

GAA does not specify client implementation details. A possible implementation of GAA client on mobile phone can use Java ME if the device manufacturer supports JSR 177 specifications. Because with JSR 177 support, the GAA client can communicate with smart card applications for credentials. On the other hand, the GAA client could be native client but provides a java interface to AP client (see Figure 3.4) for accessing credentials.

3.5 Mobile Phone as Identity Tool

Ability to perform cryptographic operations securely (e.g., in SIM card) enables the hand-held devices to perform digital signatures. Indeed, digital signatures by mobile devices termed as *mobile signatures* play a central role in mobile commerce today. In this Section, we first discuss mobile signatures and see how they enable devices to serve as identity (possibly multiple) tool.

3.5.1 Mobile Signature

European Telecommunication Standards Institute (ETSI) defines the term "Mobile signature" as follows [20]:

Mobile Signature. *It is a universal method for using a mobile device to confirm the intention of a citizen to proceed with a transaction.*

ETSI has made available the technical specifications for mobile signature framework. The framework enables mobile signature as a service for users and application providers (whose services users might be interested in). Using this service users can sign data or transaction with non-repudiable signature. The signature framework consists of 3 main components- Application Provider (AP), Mobile user, and Mobile Signature Service Provider (MSSP). The communication channel between components is Internet enabled and Mobile Signature Service (MSS) is provided as as a web service. ETSI specifies XML encoded protocol messages between AP, MSSP and mobile client.

Mobile Signature Modes

MSS supports different messaging modes namely synchronous mode, asynchronous client-server mode and asynchronous server-server mode. Different messaging modes signifies if the transaction can be proceeded in synchronous mode or not by MSSP. A transaction can not be proceeded for example in synchronous mode if the mobile users in not connected or mobile user in not ready to confirm the transaction for whatever reason.

For such cases, asynchronous mode is provisioned where AP can receive the status of a transaction signature later from MSSP after the user has signed or rejected the transaction. Asynchronous mode again depends on how MSSP behaviour is defined. For instance, MSSP can work in one way messaging system where the AP needs to request MSSP for the status of transaction. Alternatively, MSSP can be more active and could inform the status of transaction (signed or not signed) to AP as soon as the transaction is processed by user. The former mode is named as "Asynchronous-client-server mode" and the later is named as "Asynchronous- Server-Server mode". An additional advantage of asynchronous mode is that MSSP also provides value added services in terms of storing the signed transaction details.

Mobile Signature Quality

End users may not have same quality of signature capabilities. The quality (strength) of signature depends on type of device (e.g. temper resistant card, hand-held memory) used for signature creation, cryptographic capabilities of device (e.g. key length of algorithms used) used among others [20]. It is also possible that users want different signature strength depending on nature of transactions they are making. It is therefore essential that MSSP provides different signature capabilities and associate the services with quality of signature. For instance, a non-repudiable signature requires that signature is generated inside temper resistant storage.

MSSP facilitates varying quality of signature support through "Mobile Signature Profile". The profile among other fields contains mobile signature policy and description of technical protection of credentials such as where the private keys should be stored, and what algorithms are used for signature.

Multiple Identities

A mobile device identity tool requires two primary components. A temper resistant secure element (such as SIM or other smart cards) and a front end application (identity proxy) to interact with secure element for credential store and cryptographic operations.

Since these components are generic in nature, there is no restriction in terms of "identity profiles" that can be supported. There are in fact genuine use cases for mobile user to maintain multiple identities. For instance, to reflect multiple roles assumed by same person in different capacities namely administrator of a department, employee of an organization and as a personal user.

A possible implementation of multiple identities in hand-held devices is based on "pseudonymity pool" [26]. This pool is basically a collection of several identities. Each entry in the profile contains pseudonymity type, pseudonymity content and pseudonymity identifier and uniquely defines a (pseudo) identity.

Requesting an identity profile in mobile device is not trivial as the IdP (issuing an identity profile) needs to verify the user. This can be achieved by a "profile loading certificate" which the user produces to verify himself. Profile loading certificate in the device could be installed in several ways. It can be for example be a pre-installed certificate to user by hand-held operator. Another possible method is that user first receives a certificate from home IdP and uses this certificate to verify for the new identity. CLIP solution for online certificate in an organization (discussed in Section

3.4.1)can be used in the later method.

Using mobile phones as a identity token to use with PC is investigated by Abe et. al [10]. The mobile phone contains private key in the SIM which provides a security assertion signed by private key for identification. The solution intends to replace the current usage of hardware tokens as an authentication tools by mobile phones.

In this solution, mobile user needs to authenticate himself to the mobile phone (using pin code or some biometric method). After successful authentication, mobile phone issues a SAML assertion signed by private key on device SIM. This assertion is later sent to service provider for authentication. The overall procedure can be summarized as follows. When the user requests access to a SP service, the SP redirects the user to IdP on mobile phone (the solution describes a mechanism of discovering URI of mobile phone IdP). The user, if not authenticated to mobile device already, authenticates himself using pin or biometric authentication. The SAML assertion as an authentication response is sent to SP using HTTP POST method. After successful verification of response, service is granted by SP.

SIM storage provides usability as credentials are not tied with mobile phone and changing mobile device should not restrict the service.

3.5.2 Short-lived Certificates in Mobile Systems

PKI in mobile systems poses different set of challenges [18]. Authentication of Gateway Servers such as WAP servers by mobile sets was a major challenge in 2000s . This was primarily because storage of CRLs as well as computation for validation of certificate chain was not feasible in mobile sets which possessed low computation power and low storage capacity. Short-lived certificates was proposed as a means of server authentication by WTLS version 1.2 called WTLS1.2 to overcome this challenge. Short-lived certificates to WAP gateway by CA on a daily basis. WAP gateway uses this certificate for client session on that specific day. If the private key of WAP gateway is compromised, CA simply stops issuing short-lived certificates thus eliminating the need of CRL.

In recent years, there has been a significant improvement in computation and storage capacities of mobile devices. Storage of certificates as well as revocation checks are not a limiting factor for mobile devices. Consequently, the concept of short-lived certificates as an authentication method can be extended to mobile users to benefit in several scenarios. [41] describes use cases where a mobile user can provide its identity in disconnected scenario by means of short-lived certificates. When connected, the mobile

user initializes its identity from Shibboleth , which stores the credentials for this mobile user. The mobile user receives a short-lived certificate (as defined in Section 2) and uses this short-lived certificate to provide its identity in disconnected scenario.

3.6 Major Challenges

Secure storage of private keys in mobile devices is an active research area [18]. On one hand, hand-held based solutions provide more flexibility in terms greater availability of storage and computation resources as well as rich set of support for applications. On the other hand, SIM based solutions offer higher level of security as the private keys never leave the SIM but provides less flexibility in application support. Also, SIM based solutions have restricted storage and computing resources.

Online registration and issuance of certificates is another challenging area in mobile authentication [35]. This requires some form of user credentials availability (transparent to mobile client) with CA. However, there is no industry-level agreement for procedures of online registration and issuance. Further, though offline credentials are available with mobile user's home identity provider (such as National Identity Providers), lack of interoperability agreements and functionality limit the access of these credentials to home environment only.

Chapter 4

Requirements

This Chapter describes the requirements to fulfill the research objectives as outlined in Section 2.2. However, we begin with discussing usage scenarios which are relevant to the research problem and offers general guidelines for requirements. Further, we define the requirements in terms of criteria which cover usability and technical aspects of the solution. These criteria are also used to evaluate the quality of solution using Goal Question Metric (GQM) approach [13]. Later in Chapter 7, we analyze the solution against each criteria defined in this Chapter.

4.1 Usage Scenarios

In all the use cases we consider, a mobile user requests and receives a short-lived certificate while inside home network and uses the certificate for authentication/service access possibly outside home network wherein the mobile user may not be connected to home IdP. We focus on user's requirements and leave the technical details for subsequent Chapters.

4.1.1 Mobile Traveller

We first consider a use case where a mobile user is travelling outside his home network. The scenario is presented below:

An organization implements an IdP to authenticate for services within the home network. However, some services (such as access to a software product database) are not available to users from outside home network for security reasons. If acquiring short-lived certificate within home network is through strong authentication mechanism, the

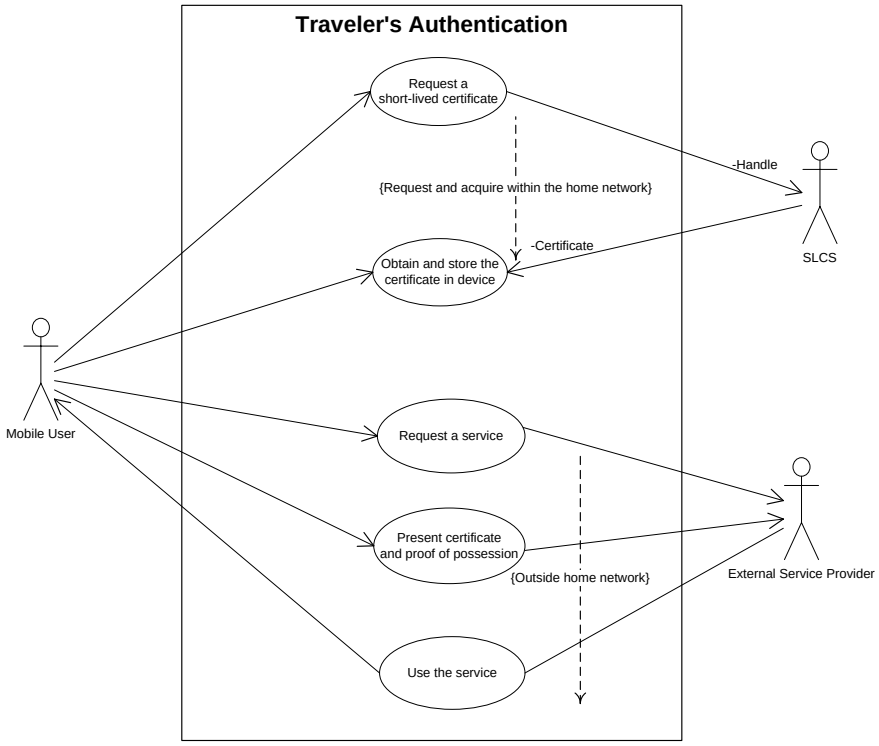

Figure 4.1: Authentication for Services Outside Home Network

access of services (using acquired short-lived certificate) while roaming can be considered strongly authenticated. The mobile traveller acquires a short-lived certificate before leaving home network. While on move outside home network, the traveler presents the certificate and proof of possession of private key to authenticate for services.

Furthermore, the same scenario can be used for accessing third party services if the third party trusts the certificate issued to mobile user. The user or SP need not contact user's IdP for authentication.

Use Case Description

Figure 4.1 illustrates the scenario. We discuss the details of each use case in following:

Request Short-lived Certificate. The user requests for a short-lived certificate while inside the home network. Several mechanisms are possible for online certificate request [34, 41]. For instance, a simple mechanism could be successful log-in to user's account from within the home network and request a certificate which is provided as service within the organization [39].

Obtain and Store the Certificate. There must be a mechanism to obtain the certificate on-line and stored securely. Section 3.3 refers to several existing secure storage solutions in mobile devices.

Request a Service. User can make a service request either to to his own organizational service or to a third party SP if there is a trust relation between third party and certificate authority used for granting the short-lived certificate.

Certificate and Proof of Possession. Either certificate or a URI indicating the location of certificate can be produced to the SP. For the proof of possession of private keys, several mechanisms (see Section 3.5) can be applied.

Service Usage. The SP grants access and usage of service after verifying the certificate and proof of possession of private key.
The scenario as discussed above can be extended to include following use cases:

1. *Disconnected Scenario:* Mobile traveler can be considered as special case of more generic disconnected scenario because traveler need not contact organization's IdP while accessing services. Node to node interaction is also possible as either node need not be connected with IdP. Both nodes exchange their certificates and communication proceeds.

2. *Single Sign-on:* Short-lived certificates can be used as a single sing-on for the duration of certificate validity. If the security requirements do not demand temper resistant storage and pass phrase protection, the credentials can be kept in file system. The applications would not require user's intervention for pass phrase or pin entry to access the private key thus facilitating Single Sign-on. For strong security needs single sign-on (as suggested above) is *not* recommended and credentials must be pass phrase protected and stored in temper resistant storage.

4.1.2 Mobile Signature Service

A mobile device in secure possession of a short-lived certificate can be used in mobile signature systems (see Section 3.5.1). We consider following use case scenario.

A mobile user intends to use WLAN service in an airport lounge using his laptop. The user first connects to WLAN portal where AP is asked to enter his mobile number. The user receives a SMS containing text that need to be signed to gain access. The user

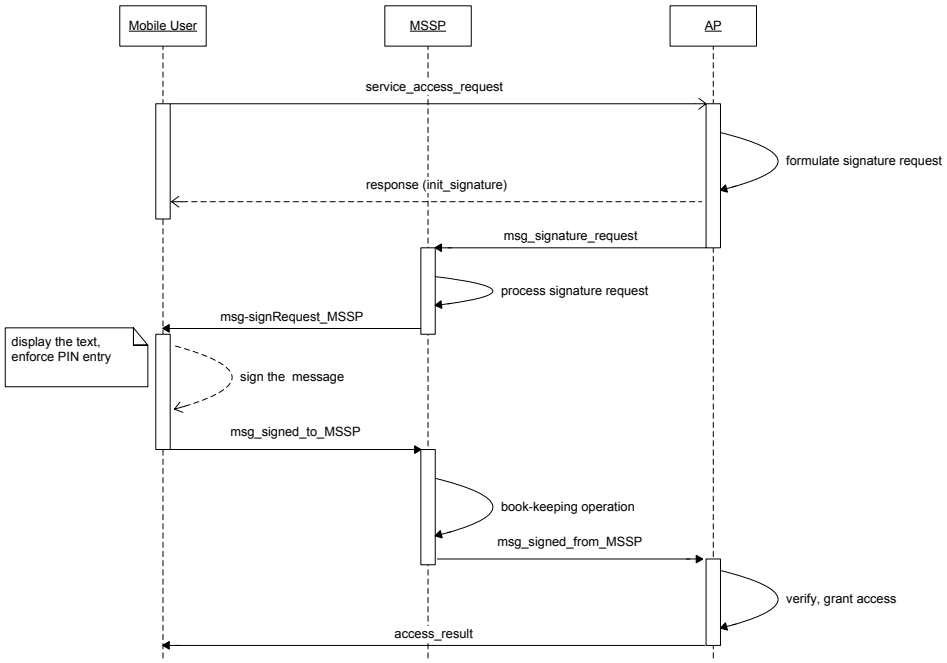

Figure 4.2: Message Signature Service. Sequencing of messages among components

signs the text (usually by pressing OK and PIN validation) using short-lived certificate and is granted access of WLAN.

Following is a step by step process of the scenario. Figure 4.2 depicts a sequence of activities.

1. The AP receives WLAN access request in HTTP browser from user which contains user's mobile number.
2. CP uses mobile number to formulate a message request for MSSP.
3. AP also informs user that a signature service request is being invoked.
4. AP calls MSSP's mobile signature method to submit the signature request intended for the user.
5. MSSP evaluates the message and decides a mode of transfer of the message to mobile user.
6. MSSP sends the message to mobile user which is displayed on the handset.
7. User confirms or rejects the signature request. In case of confirmation, the private

key of short-lived certificate is used to sign the messages.

8. The signed message is dispatched to MSSP by mobile device.
9. MSSP verifies the signature, performs book-keeping activities (such as logging of request, time-stamping) and signs the response using MSSP's private key.
10. MSSP dispatches the signed response back to AP.
11. AP verifies the response and provides WLAN access to user.

MSSP, among other possibilities [20], can either be an independent third party or can be deployed by service provider or can be provided as service by user's IdP. The only requirement from the usage of our short-lived certificate perspective is that there is a trust relation between MSSP and mobile user as well as between MSSP and AP.

The short-lived certificate usage in MSSP is particularly beneficial for mobile travellers (see Subsection 4.1.1). The traveller can obtain a certificate before leaving home network which can be used for signature services for the (short-lived) duration of certificate.

If the signature obtained for mobile device is to pass the criteria for *qualified signatures*- a signature considered to be equivalent to handwritten signature as defined in the directives of 1999/93/EC, a framework for electronic signatures, key generation and storage of private key must be performed in resistant temper proof smart cards.

4.1.3 Mobile Phone as a Security Token

Our solution can be used as security token which prevents password leakage when used with a PC. The scenario can be described as below (see figure 4.3):

The user wants to access a service and makes a request via HTTP browser of PC. The SP shows a list of trusted IdP in the browser and user selects an IdP which corresponds to the short-lived certificate the mobile device carries. Afterwards, user is prompted for the user-id associated with the IdP but not the password. SP generates a redirection URL from the combination of IdP and user-id and redirects[1] the browser to mobile phone. The mobile phone acting as IdP server, prompts at the display if the user wants to send credentials. If the user wishes so, it authenticates itself with a PIN entry and SAML assertion signed by private key of short-lived certificate is sent to PC browser which in turn sends the assertion to SP. As the SP possesses root CA

[1]To resolve the URI of mobile set, a relay server can be used. See Abe et *al.* [10]

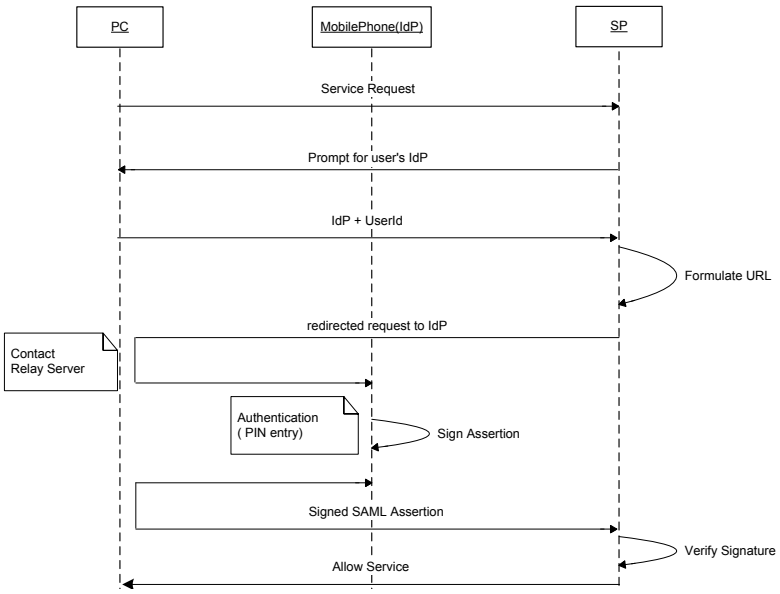

Figure 4.3: Mobile Phone as Identity Provider

certificate for the short-lived certificate on mobile phone, SP can verify the signature of the response. Successful verification results in granting of service.

A significant benefit in the scenario is that PC users need not enter password in browser.

4.1.4 Lessons from Usage Scenarios

The use cases covered in this Section bring several challenges for a feasible real world solution. For instance, in case of mobile traveler there must be a short lived certificate service from home organization to issue short-lived certificate real time and online. Further, the private keys must be generated and stored securely in mobile device to avoid compromise of private keys. It is also important that mobile device provides functions for signature verification, message signing and functions to participate in challenge-response messages from service providers or another mobile node. These messages are needed for the proof of possession of private keys.
In following Sections, we formally lay down the requirements for solutions in terms of usability and technical criteria.

4.2 Usability Requirements

Usability criteria concerns with requirements that should be fulfilled from user's perspective. These criteria in general assess user friendliness of solution which should include platform independency and ease of software installation for solution to work.

Criterion 1 *The solution should be easy to use.*
Certificate request needs to be initiated and confirmed by user. It is important that request is clearly displayed on screen and include information about CA. The user must also be provided with option to confirm or reject the certificate before dispatching the request. In general all prompts should be clearly displayed to the user. Further, the delay in authentication procedure should be within tolerable limit (e.g., 0-60 seconds) for users. This criteria also contributes to commercial viability of solution.

Criterion 2 *The solution should be platform independent.*
The solution should not be limited to a particular platform (or a set of platforms), rather it should address the thesis problem in a generic way.

4.3 Technical Requirements

Criteria 3 *Key pair (public and private keys) should be generated inside the card.*

Functionality to generate key pairs is crucial for safety of private key. Moreover, several signature standards [20] demand key pair generation inside temper resistant card.

Criterion 4 *Private keys should be securely stored.*

Compromising private key defeats the whole effort put into authentication of mobile user. Section 3.3 discusses the importance of securing keys. Further ETSI signature standards demand that private keys should never leave the temper resistant card. The solution therefore should ensure that private keys do not leave the device.

Criteria 5 *Short-lived certificate request and issuance (to mobile client) should be online and real-time.*

The usage scenarios discussed in Section 4.1 are useful when the request and issuance process can be performed in real time. For instance, mobile traveler would not find it very useful to obtain certificate to be used in a short trip if the process of obtaining certificate is itself time consuming. Moreover, online certificate request and issuance is projected as major advantage in the thesis work over conventional off-line certificate request, verification and issuance.

Criterion 6 *Support for cryptographic operations using certificates*

For the authentication, not only user needs to provide the certificate, user has to provide proof of possession of private keys to service provider (verifier). This requires ability to sign texts using private keys. CSR also requires that certificate request is signed by private key. Another requirement is verification of digital signature. This is required during mutual authentication (mobile user verifying service provider's certificate or another mobile user's certificate)

Criterion 7 *The solution should be scalable.*

The solution should be scalable to growing number of mobile users and CAs.

Criterion 8 *Use of open standards in the solution.*

Open standards ensure that solution is scrutinized to maximum detail thus helps realize the security loopholes in the solution. Also, open standards make interoperability issues easier to implement.

4.4 Evaluation Metric for Requirements

GQM is a widely used mechanism for requirement evaluation, quality control and feedback in software projects [13]. We apply GQM mechanism by listing set of questions (Q.n.m) or goals required to fulfill each criteria and matrices (M.n.m) in form of associated quality matrix. A matrix as "binary" means that answer can be given in Yes or No. When we have a matrix in form of "Binary, Description" it means that a description needs to be associated with binary answer for explanation.

Criterion 1 The solution should be easy to use.

Q.1.1 Are there any software installations required?
M.1.1 Binary, Description
Q.1.2 What are the user steps in registration process?
M.1.2 Description
Q.1.3 Does the authentication process require user intervention?
M.1.3 Binary

Criterion 2 The solution should be platform independent.

Q.2.1 Are the solution features applicable to all the platforms?
M.2.1 Binary, Description

Q.2.2 Are there any special needs for client software?
M.2.2 Binary, Description

Criterion 3 Key pair (public and private keys) should be generated inside the card.

Q.3.1 Are the key pair generated inside the card?
M.3.1 Binary, Description
Q.3.2 What key lengths are supported?
M.3.2 Key-length in bits, Description

Criterion 4 Private keys should be securely stored in device

Q.4.1 What private key storage options are supported in the solution?
M.4.1 Description
Q.4.2 What are the security mechanism deployed for private keys?
M.4.2 Description
Q.4.3 Are external smart cards supported in the solution?
M.4.3 Binary
Q.4.4 What are special requirements for using external smart cards?
M.4.4 Description

Criterion 5 Short-lived certificate request and issuance (to mobile client) should be online and real-time.

Q.5.1 What is the average time for making a short-lived certificate request from mobile device?
M.5.1 Time in milliseconds
Q.5.2 What is average time duration from the moment certificate request is made and a certificate response received?
M.5.2 Time in milliseconds

Criterion 6 Support for cryptographic operations using certificates

Q.5.1 What cryptographic operations are supported by the solution?
M.5.1 Description

Criterion 7 The solution should be scalable.

Q.7.1 Are there any restrictions on scalability of solution?
M.7.1 Binary
Q.7.2 Is the performance affected in presence of multiple certificates and keys?

M.7.2 Binary

Criterion 8 *Use of open standards in solution.*

Q.8.1 Does the solution adhere to open standards?
M.8.1 Description.

4.5 Criteria Priority

Table 4.1 summarizes the criteria and associated priority in the solution. Lower number denotes higher priority.

Table 4.1: Criteria Priority

Criterion	**Description**	**Priority**
1	Easy to use solution	1
2	Platform independent solution	2
3	Secure storage of private keys in device	1
4	Key pair generation inside the card	1
5	Online certificate request and issuance	1
6	Cryptographic operations support	1
7	Scalability of the solution	1
8	Use of open standards in the solution	2

Device independent solution may not be feasible as many mobile set manufacturers do not support device independent standards to full scale. Especially, communication between SIM and external applications (Midlets) are not supported in many existing devices in market. Accordingly, we have associated a lower priority with criterion 2. Scalability, usability and secure storage of private keys are crucial for the real world usage of the solution and hence been assigned highest priority.

Chapter 5

Solution

This Chapter provides solution to the research problem described in Section 2.2. Research problem consists of two parts, namely interaction between the CA and mobile set to obtain online certificate and secure credential store for storing private keys. We first provide a general approach to overall solution which combines both parts and later discuss two components separately in detail.

5.1 Solution Approach

The design of credential store can be considered separately from interaction between mobile set and CA. With a well defined interface, credential store can provide services for credential store as well as functions such as CSR and signature generation which uses private keys so that they do not leave credential store. Interactions with CA can be conceptualize in a separate component which handles communication between CA and mobile set as well as uses credential store interface for operations requiring private keys. We introduce component "Credential Manager (CM)" and "Client Proxy (CP)" to address credential store and interaction between mobile set and CA respectively. Figure 5.1 depicts a basic diagram of components and their interactions. To understand the roles played by three components we discuss the case of online certificate request. The steps are follows:

1. CP informs CM to build a CSR.
2. CM generates CSR (after creating a key pair) and returns the CSR to CP.
3. CP sends the CSR to CA.
4. CP receives the certificate.

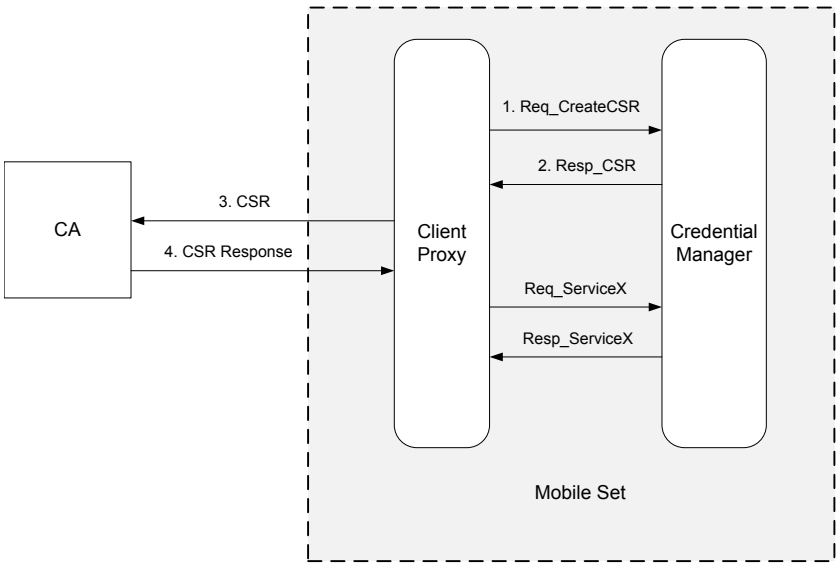

Figure 5.1: Components of the System

Figure 5.1 also shows generic service request and response between CP and CM.

Several details have been left out at this stage such as key length, choice of algorithm, storage medium and security of private keys. Nevertheless, the interaction provides a preliminary structure of the proposed solution. As CP requires CM's services, we define an interface named "Credential Manager Interface". The interface is implemented by CM and used by CP. The Interface abstracts the details of credential store and also encapsulates the storage medium into the Interface implementation.we define each component as follows:

CA Service. *A short-lived credential service (SLCS) which accepts CSR from mobile client (through CP) and provides short-lived certificate to mobile client if request is successful.*

The CA requires a mechanism to verify the CSR from client. There are many implementations possible including off-line credentials availability [39] and trust relation with Federated IdP [41]. However, we take up the issue of request verification mechanism deployed with CA when we deal with the SLCS implementation details.

Client Proxy. *A component that implements SLCS client to interact with SLCS service at one end and uses CM Interface to perform operations requiring private keys on the other end.*

As outlined in Section 5.1, CM is responsible to provide a transparent interface to CP.

Credential Manager. *A component that implements credential store and provides credential management abstraction to CP through a well defined interface.*

Service Based Design

We present the design of components envisaged in Section 5.1), in terms of specifying services the component offers and how these services can be conceived and implemented. Design of each service is presented under headings *Input* (parameters the service accept), *Processing* (how the service is implemented), *Restrictions* (in terms of input values or system) and *Output* (the actual service result). This service design description is useful because:

1. A service might comprise of several code functions whose details can be well described in implementation.
2. The service encapsulates the complexity and communication among services can be achieved through well defined interfaces

Following Sections discuss the design of each component including services offered and tasks performed.

5.2 Credential Manager

One of the objectives of CM is to offer a choice in different levels of security requirements for the credential store. In one extreme, the credentials can be stored as plain file in the device (such as Java ME records). On the other extreme, credentials can be stored in temper resistant smart card for highest security needs (such as payment transactions). The different security levels are made available through concept of Security Element (SE). SE is a combination of following:

- *An SE device* which stores the credentials and perform cryptographic operations on them. The SE device, for instance, can be fully implemented in software or can be implemented in a smart card.
- *A Security Policy* specified and implemented by SE. The policy includes information such as key length support and user prompt for pass-phrase.

5.2.1 Technology Selection for Credential Store

Several technology alternatives are discussed and presented in Section 3.3. We select Java ME based SATSA for following reasons:

1. Java ME application support is provided by all the major manufactures (e.g. Symbian, Windows OS) in market.
2. SATSA, as an extension of Java ME supports APDU protocol for communication between device and smart cards (e.g. SIM).
3. SATSA PKI package supports signature generation inside smart card.
4. SATSA supports generic security elements (e.g. external smart cards, security elements entirely implemented in software)
5. Open Mobile Alliance mandates the support for SATSA-PKI and SATSA-APDU features in all mobile sets [26].

Furthermore, the choice fulfills the requirements Criterion 4.1.2 (platform independent solution) and Criterion 4.2.8 (use of open standards in solution) as defined in Chapter 4.

5.2.2 Architecture and Services

Selection of SATSA provides an extensive set of APIs over Java ME for storage and usage of credentials. SE integrates directly with SATSA as an argument in APIs. CM implements one or more SEs (or uses available SEs). For an operation involving credentials, CM first selects an SE and calls a typical SATSA function for actual operation. The selection of SE is determined by configuration parameters[1] (see figure 5.3).

CM provides a crucial abstraction in terms of implementation of SEs. Furthermore, if the security requirement is low, the credentials can be stored in plain text. Storing the credentials in plain text allows applications on a mobile device to access credentials without any need of pass-phrase or PIN entry. A significant advantage of such arrangement is to facilitate single sign-on for service access. CM implements the functionality of plain text storage as well and offers this feature (for low security requirements) if indicated so in configuration parameters. Figure 5.2 illustrates the role of SATSA to achieve the objectives of CM. In diagram, SE is implemented in SIM using a Java Card

[1]The parameters may be supplied as a configuration file or record (depending on implementation) and are read by service during startup. Prototype implementation provide details of parameter fields and associated values.

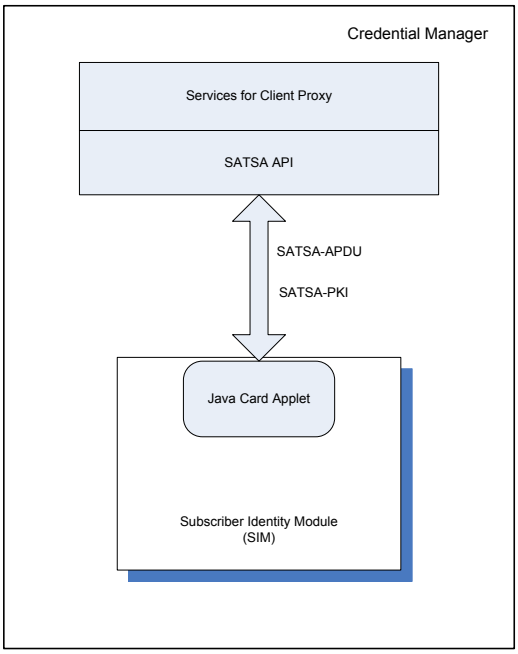

Figure 5.2: Credential Manager Architecture

applet, of course this does not limit the choice of SE which could be for instance, an external smart card.

The services offered by CM are as follows:

1. Generate CSR - The CM accepts parameters required for CSR and returns a generated CSR to CP
2. Credential Add and Remove - The CM provides add and remove certificate service from SE if required parameters and authenticity (e.g. pin verification) is supplied by CP.
3. Signature Generation - Generates a signature for CP. The CP informs CM if the signature is needed for authentication or for non-repudiation.
4. Signature verification - Accept signature verification request and return the result.
5. Auxiliary service - Listing of credentials.

Generate CSR

Input

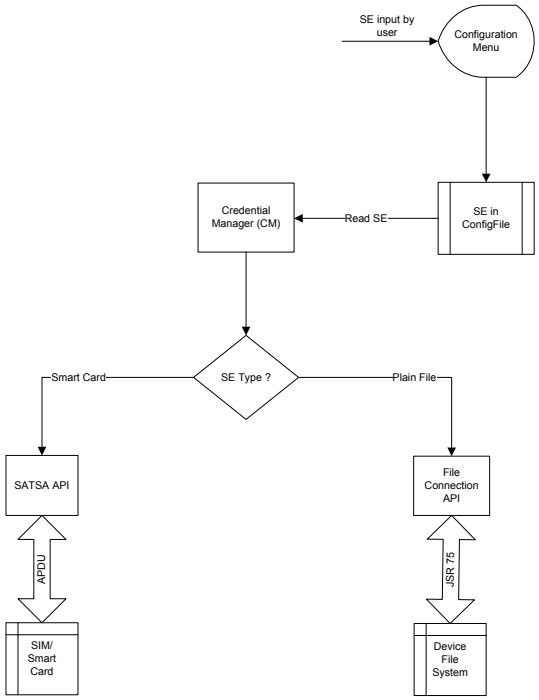

Figure 5.3: Selection of Security Element for Credential Store

1. DN of the new certificate (*optional*). Depending on implementation, the DN might be determined and supplied by CA.
2. Public key algorithm for the certificate. The algorithm could be for example DSA or RSA.
3. Key Length - Key length of the algorithm.

Restrictions

CM must ensure that user approves the CSR (e.g. using PIN entry) before proceeding to handle the request. If the user rejects CSR or PIN entry is not correct, the generation is discarded and an exception is thrown.

This is a synchronous service and call to CSR is blocking.

Processing

For the implementation, service first selects SE from configuration parameters. SATSA provided `generateCSR`[2] function is called with selected SE for actual operation. However, key generation is forced and input parameters are checked against a configuration parameters before passing on to `generateCSR`. The actual CSR generation is performed by SATSA-PKI.

Output

Generates a certificate request in PKCS#10 format. In case of restriction violations, a null is returned.

Credential Add and Remove

Input

1. A user friendly name associated with certificate. CM uses this name to refer to the certificate (e.g. while requesting removal of a certificate from credential store of CM)
2. For certificate add - The full certificate path (i.e. it should contain user certificate and issuer certificates in certificate chain)
3. For certificate remove - User friendly name of the certificate as used for certificate store, issuer and serial number of the certificate.

Restrictions

The addition and removal of certificates to CM are authorized by user with a PIN.

Processing

The CM provides this service through two API calls, one each for certificate add and delete. CM calls SATSA-PKI add and remove credential functions after determining the SE from configuration parameters. The Security element requirement for SATSA-PKI is determined from configuration parameters. Exceptions generated in SATSA-PKI are simply forwarded.

Output

The Boolean result indicating success or failure of operations.

[2]This function is part of `UserCredentialManager` class in SATSA-PKI package.

Signature Generation

Input

1. DN of the certificate to be used for signing.
2. Byte array to be signed.

Restrictions

The signature generation must be confirmed by user with a PIN entry.

Processing

Service calls SATSA-PKI `CMSMessageSignatureService` object function with additional parameter for SE which is read from configuration parameters.

As per `CMSMessageSignatureService`specifications, null entry means that user selects a certificate from the list of certificates provided at display. Exception generated by `CMSMessageSignatureService` are simply forwarded.

Output

The signed message is returned in a byte array.

Signature Verification

Input

1. The signature text to be verified.
2. The message associated with the signature.
3. Sender's certificate.

Processing

Service first extracts the public key and signature algorithm from the input certificate. Then it calls a series of SATSA-Crypto functions to verify the signature as follows. A signature object is instantiated using signature algorithm extracted in previous step. Verification is initialized using sender's public key. and original message is fed to the Signature object using object's update method. Finally, the input signature is compared in verify method of Signature object. The verify method returns a Boolean (true or false) indicating verification result.

Output

Signature verification result in Boolean.

5.3 Client Proxy

Client Proxy is responsible for communicating with CA to request and obtain short-lived certificate. At this point we assume a SLCS server (see Section 5.1 facility offered by CA to interact with CP and issue short-lived certificate. The issuance of certificate is two step process. In first step, CP sends a log-in request to SLCS and receives authorization response after successful log-in. The authorization response is used to generate CSR by CP. In second step, CP dispatches this CSR to obtain a certificate in response. At one end, CP implements HTTP client to communicate with SLCS server (and thus CA). On the other end, CP uses CM's service interface to store and use credentials. The messages between CP and SLCS are exchanged as XML messages. We provide the syntax and details of these messages as we discuss the related activities. A sequence of activities for CP is presented below.

1. CP requests a log-in to SLCS service to obtain authorization response.
2. CP parses the authorization response to obtain parameters for CSR.
3. CP uses CM's interface to request CSR generation by supplying CSR parameters obtained in previous step.
4. CP sends CSR to SLCS service and obtains short-lived certificate as CSR response.
5. CP calls CM to store the certificate.

We discuss design steps for each activity in detail.

SLCS Service Log-in and Authorization Response

Authorization response from SLCS server is achieved through a series of steps. The first step creates a HTTP Client (or simply client) object by passing user-id, password and URI of SLCS service. The information is obtained from a configuration file which lists these parameters. For now, we assume an external service available to CP which redirects the client to the native IdP. CP's responsibility at this stage is to successfully authenticate with IdP. CP receives a handle in response which is forwarded to SLCS. The handle is processed with SLCS which results in an authorization response from SLCS to CP. This authorization response contains information required for CSR generation. The details of authentication procedure with native IdP may vary and

depends on arrangement between IdP and mobile client. We return to the details of authentication procedure with IdP when we discuss implementation and use case scenario.

Parsing Authorization Response

If the CP could successfully log-in to native IdP, the SLCS service returns an authorization response to CP. This response is used to extract parameters necessary for CSR.

The response constitute following fields:

- <`AuthorizationToken`> This string uniquely identifies a session between SLCS server and client.
- <`SubjectDN`> The certificate must use this value as DN.
- <`EncryptionAlgorithm`> to be used in the certificate.
- <`X.509Extensions`> one or more extension attributes (in form of attribute name =value) for X.509 permissible extensions.

The implementation defines length of each of the fields and values of the authorization response which is necessary for correct parsing of the authorization response.

CSR Request to CM Interface

Parsing the authorization response provides following parameters for CM's Generate CSR service (see Section 5.2)

1. DN
2. Encryption Algorithm
3. X.509 extensions

It is CM's responsibility to generate a CSR and return to CP.

CSR Message to SLCS

The CP after receiving CSR response (from CP), performs following activities:

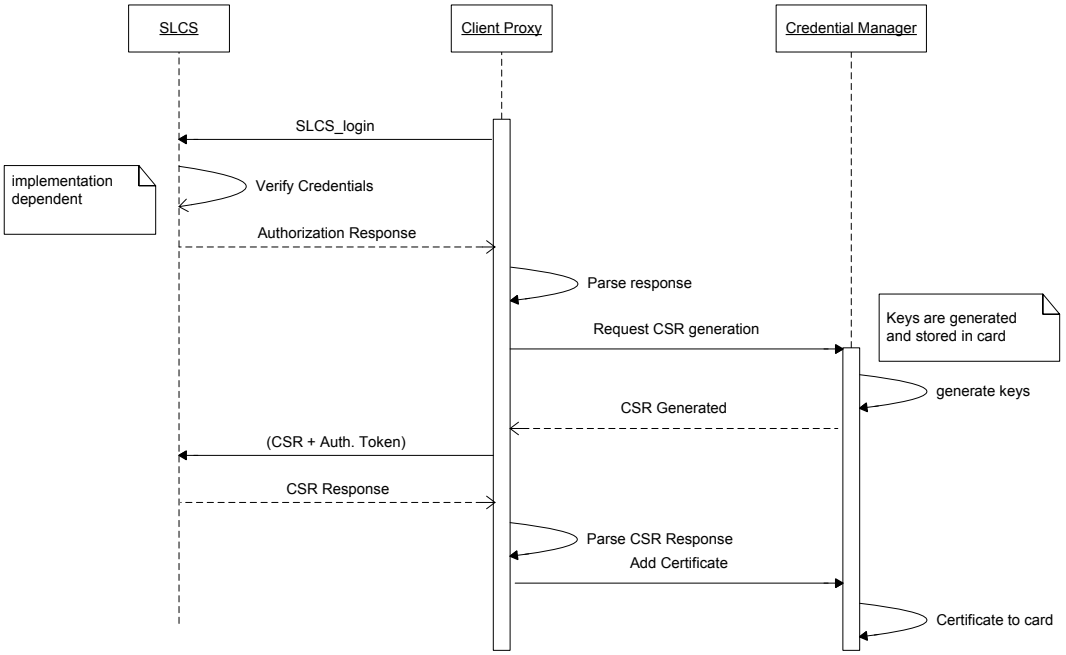

Figure 5.4: The process of login, authorization response, CSR formation and finally certificate as CSR response

1. Encoding of CSR response, if required by SLCS[3].
2. Append the extttAuthorizationToken (of current session) to completer the message.

The details of the fields are as follows:

- <`AuthorizationToken`> Belongs to current session, extracted from log-in response message.
- <`PEMEncodedCSR`> The response of CSR request to CM.

[3]SLCS is considered an "external" service and details of encoding requirements depend on actual SLCS implementation. The prototype implementation (Chapter 6) uses PEM encoding for CSR and Certificate messages.

CSR Response from SLCS

SLCS verifies CSR message (such as matching authorization token, see Section 5.3) received from CP. If the verification is successful, SLCS issues a CSR response which contains the short-lived certificate.

CP forwards the certificate to CM after parsing and extracting the certificate from the XML message.

5.4 SLCS Service

Although CP interacts closely with SLCS, we have so far not discussed the design of SLCS and how SLCS performs the activities of credential verification and issuance of short-lived certificate to CP. Since SLCS is a service essentially offered by CA, the actual implementation such as interaction with CA to obtain certificate and interaction with IdP of SLCS client to obtain credentials can vary greatly. This Section therefore discusses generic features and services SLCS must provide.

Following are the major activities of SLCS:

1. Listen and accept the incoming log-in requests from SLCS clients.
2. Interact with client's IdP to receive client's credentials.
3. Formulate and send a log-in response to CP based on previous step.
4. Accept and verify CSR from CP.
5. Interact with Online CA to send the CSR and receive the short-lived certificate online.
6. Forward the certificate to CP.

A prototype implementation of SLCS is described in Chapter 6.

Chapter 6

Prototype Implementation

This Chapter describes a prototype implementation of the solution discussed in Chapter 5. After an overview, we address implementation of each component of the system and focus on how the services offered by components are realized.

6.1 Implementation Overview

As described in Chapter 5, the proposed solution comprises of 3 components namely SLCS, CP and CM. SLCS is external to the device and essentially functions as RA since it is responsible for verifying the certificate request and forwarding the request to CA for (certificate) issuance.

EGEE Grid Middleware (refered as "gLite") provides an open source implementation of an SLCS server to issue short-lived certificates to Grid users (see Literature Review, Chapter 3). This open source implementation fully serves the purpose of SLCS server for our mobile client. The Grid SLCS provides features for redirection to home IdP and exchange of SAML messages afterwards. Moreover, the Grid SLCS client opens an HTTP connection to connect to SLCS server and so does our SLCS client (CP component implements HTTP client). Although, reusing Grid SLCS server requires Shibboleth based IdP, we can live with this restriction as our prototype implementation also uses Shibboleth for home IdP.

We discussed SATSA as a technology choice for credential store in Section 5.2. CM is coded in Java ME for smooth integration with SATSA and CP. Java ME provides built-in packages for HTTP client implementation and thus is a suitable choice for CP. Both CP and CM interacts with a single Error Handling and Logging module for

central error handling and logging.

6.2 Short-lived Credential Service

As discussed in Section 5.4 we use the existing SLCS service available as free source from the gLite Middleware. However, it is important to explain the SLCS features to understand the SLCS-CP integration (see Section 5.3).

Figure 6.1 illustrates the steps involved from log-in request to certificate issuance and store. The details of each step is explained below:

1. The user sends a log-in request (through CP) to SLCS which initiates a session to obtain short-lived certificate.
2. SLCS ensures that user's IdP (provided in log-in request) is registered in IdP database and redirects the CP to the IdP for authentication.
3. User authenticates to the IdP using existing credentials.
4. After successful authentication, IdP issues a handle to the SLCS service using HTTP redirection.
5. SLCS retrieves the SAML attributes of user from IdP using the handle. The SAML attributes are used to generate the certificate DN and optional extensions to be included in certificate.
6. SLCS creates a log-in response (as XML message) which includes DN and an authorization token and sends it to CP.
7. CP issues a request to CM for CSR generation using information received from SLCS as parameters (see Section 5.3).
8. CM generates a CSR and sends it to CP (see Section 5.2.2).
9. CP formulates a CSR response to SLCS in an XML message which includes authorization token (from previous step).
10. SLCS verifies the CSR, formulates a PKCS#10 request and sends the request to online CA for signing.
11. Online CA signs the PKCS#10 request and sends the short-lived certificate as PKCS#7.
12. SLCS forwards the certificate to CP as an XML message.

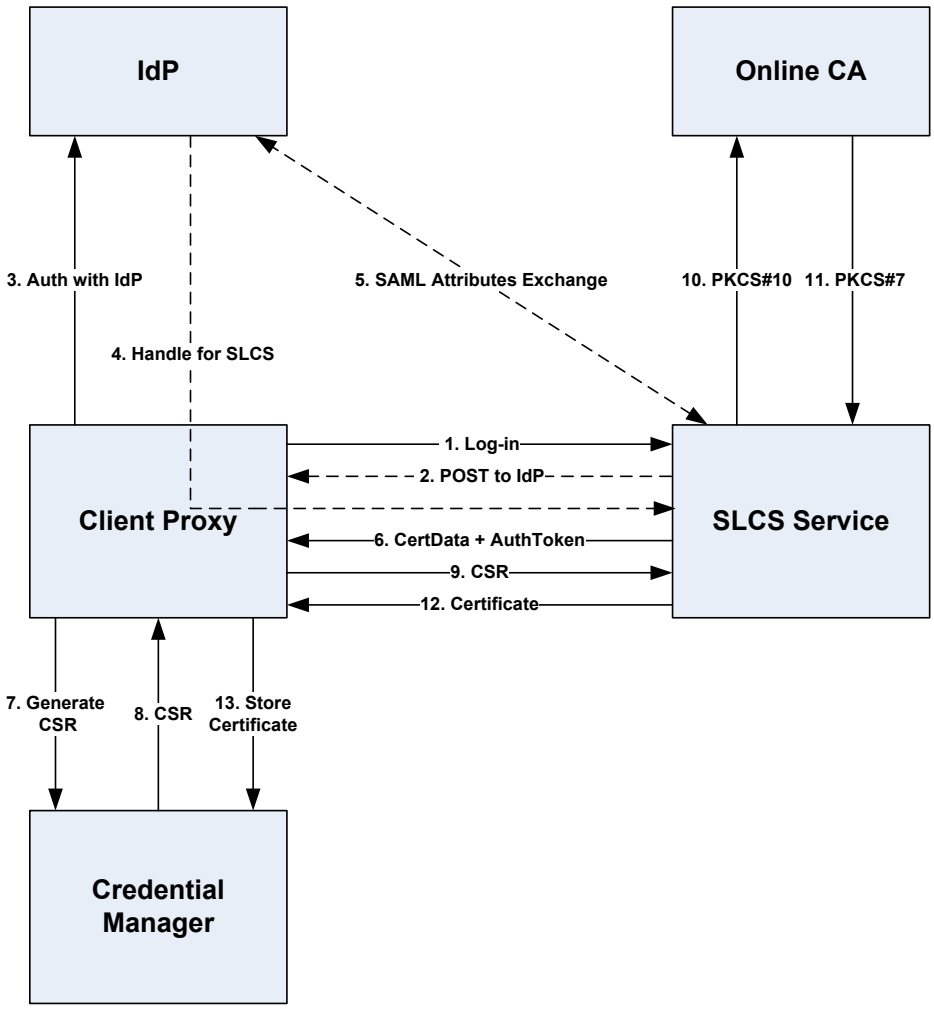

Figure 6.1: Process Flow from Log-in Request to Certificate Issuance and Store

13. CP retrieves certificate and calls CM interface for certificate storage (see Section 5.2.2).

PKCS#10 describes a syntax for certificate request to CA. The request contains DN, public key and optional set of X.509 attributes (which may be included in certificate). Similarly, PKCS#7 is a standard response message used by CA to send the certificate to SLCS.

The redirection in step 2 is standard HTTP 302 redirects as defined in RFC 2616 [3]. HTTP 302 redirects is a temporary redirect which is justified in the implementation (in step 2 and 4) as request eventually returns to SLCS.

6.3 Client Proxy Implementation

As discussed in Chapter 5, in one hand CP interacts with SLCS to request and receive certificates. On the other hand, CP interacts with CM to store certificate and credentials. Additionally, CP participates in message exchanges with service providers for the purpose of actual service authentication. We identify following implementation needs for CP:

1. Implement an HTTP client for certificate request to SLCS (implements HTTP server).
2. Calling functions to utilize CM interface for certificate and credential store.
3. Functions to interact with service provider.

CP uses an instance of CM interface (the implementation of interface is discussed in Section 6.4) to generate key pair and store certificates.

The class diagram for CP is illustrated in figure 6.2. We discuss major classes and their objectives.

MobileInit

This is a Java applet class which serves as starting point for the entire application. Function `commandAction` implements a menu driven user interface for commands.

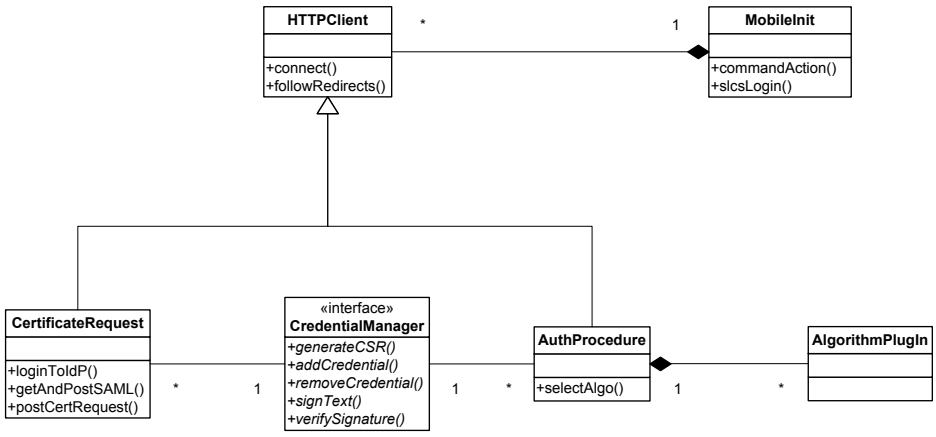

Figure 6.2: Class Diagram of the Client Proxy Component

HTTPClient

This is a generic class used for HTTP connections to SLCS and home IdP. Since these connections are needed for certificate request (connection to SLCS) and credential verification with home IdP, we create this generic class which is extended by classes (see figure 6.2) needing HTTP connections.

CertificateRequest

The objectives and features of this class are described below:

1. Login to SLCS: Connects to SLCS server (using `HTTPClient`) in order to request a certificate.
2. Follow redirections from SLCS to connect to home IdP. It calls `followRedirects` method of `HTTPClient` to achieve this purpose.
3. Authenticate with home IdP. The class uses `loginIdP` method to achieve this.
4. Parse SLCS log-in response from SLCS. The response contains authorization string to be used for next request (`postCertificateRequest`)
5. Request CM interface (using `generateCSR` method of Interface) to generate a key pair.

6. Post a certificate request to SLCS (the `postCertificateRequest` method) using authorization string and public key. The result is response from SLCS containing certificate.
7. Parse the certificate response to retrieve the certificate.
8. Call the CM interface (`addCredential` method of Interface) to store the certificate.

6.4 Credential Manager Implementation

We discussed the objectives of CM in Section 5.2. Class `CredentialManager` implements a `CMInterface` using SATSA APIs (see Figure 6.3). CM requires availability of configuration parameters before it can offer any service (see Section 5.2). The Configuration parameters are initialized in `ConfigParamater` class. The configuration parameters consists of following configurable options:

1. A table of public algorithms and associated key lengths. These are the entries supported by CM.
2. Name of the storage element to be used. "Plain text" refers that credentials are to be stored in plain file and should be accessible without pass-phrase requirement. Other values are standard storage element names. Standard storage elements names are as defined in Java Cryptography Architecture API Specification and Reference.
3. Error handling and logging parameters: These parameters defined the debug level to be used (ranging from error only to verbose) as well as name and location of logging file.

The implementation details of services are listed below:

CSR Generation

The caller of service (`CertificateRequest` class) provides parameters for service namely DN of the certificate for which CSR is required, public key algorithm to be used and bit length for key. CSR Generation (implemented in `GenerateCSR` method)consists of following sequence of operations:

1. The public algorithm and key length are checked against listed pair of algorithm and key length in Configuration Parameters to find out that the combination is

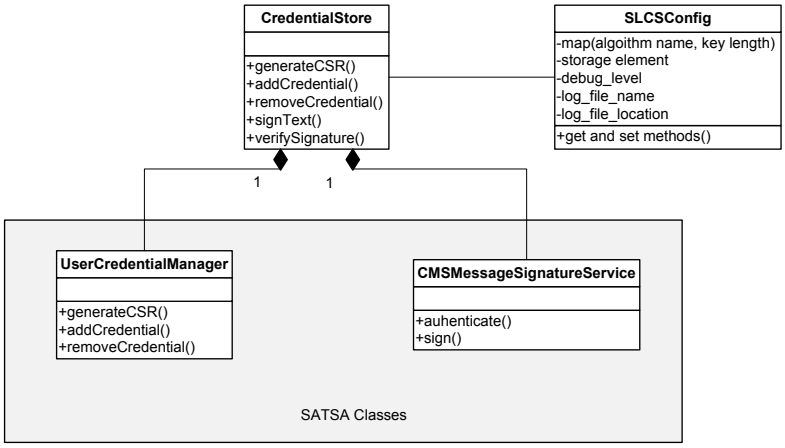

Figure 6.3: Implementation of Credential Manager Interface

valid and thus supported by SLCS application.

2. Selects a default storage element (from `ConfigParamater` class) and sets the forced key generation.

3. Calls `UserCredenialManager` (provided by SATSA) using above arguments. If the result is successful, `GenerateCSR` stores the association of public key with DN.

4. `UserCredenialManager` prompts the user to verify the request for key generation. If the user approves the request, she needs to provite pin entry for actual key generation to proceed.

5. Forwards the byte array obtained from `UserCredenialManager` to caller of the service (CP).

Add and Remove Credentials

In case of certificate add service (`addCertificate` function), caller of service provides a certificate and associated user friendly name. The name is used by caller to references the certificate for later use (e.g. deletion). The add credential operation is performed as follow:

1. Caller provides certificate to be added (as byte array) and a associated user friendly name (as String).

2. `CredentialStore` retrieves default SE from `SLCSConfig`
3. `CredentialStore` calls addCredentials function of `UserCredentialManager` using arguments from previous two steps.
4. Forwards the Boolean result to caller.

Remove credential service is implemented as follows:

1. Caller provides user friendly name (a String object) of the certificate to be deleted and issuer and serial number of certificate (see Section 5.2)
2. `CredentialStore` retrieves default SE from `SLCSConfig`.
3. `CredentialStore` calls removeCredentials function of User `UserCredentialManager` using arguments from previous two steps.
4. SE implementation asks for user PIN entry before proceeding to delete the certificate from SE.
5. If the PIN is incorrect, the deletion does not proceed and error message (see errorHandler messages for CredentialManager in Section 6.5)
6. Forwards the Boolean result to caller.

Signature Generation

The CM provides signature generation facility in SE as specified in Configuration Parameters. The implementation provides digital signature in CMS format (defined in RFC 2630 [1] and refined further in RFC 2634 [2]) which is a standard format also used by Java Cryptographic Architecture Specifications. Further, the CM supports digital signature for the purpose of authentication as well as non-repudiation. This has been made convenient because support at lower level (`javax.microedition.securityservice` package, JSR 177) exists. The implementation steps are as follows:

1. The caller provides DN of the certificate (as String), byte array [1] to be signed and also a flag to indicate if the signing is for authentication or non-repudiation. This prototype implementation only supports authentication.
2. `CredentialStore` retrieves default SE from `SLCSConfig`.
3. SE implementation displays the certificate being used for signature and prompts for PIN to user before proceeding with signing text.

[1] Credential Manager also provides function for signing text (as String objects)

4. The CM calls `CMSMessageSignatureService` class of SATSA for signature generation in SE.
5. The DER encoded signature is returned to the caller.

Signature Verification

This service is implemented as follows:

1. CP provides certificate, signature text and the message associated with the signature.
2. Public key and signature algorithm are extracted from the certificate. Java ME `javax.security.cert.X509Certificate` class implements functions to achieve this from an standard X.509 certificate.
3. A signature object is instantiated with signature algorithm in previous step (uses Java ME Signature factory method). Also, the object is initialized for verification using public key (from previous step).
4. The message (step 1) is fed to Signature object using one or several calls to `Signature.update` method until the message is exhausted.
5. Finally, the signature bytes are passed to Signature using verify function.
6. The Boolean result is forwarded to the caller.

6.5 Error Handling and Logging

The implementation provides several levels of logging ranging from error messages (`debug-level = error`) to verbose output (`debug-level= verbose`) as specified in configuration parameters. The logging is sequentially performed and saved in a plain text file using File Connection API (JSR 75). The error handler is also responsible to display the error and warning messages to user. Moreover, the handler is configured to receive and process commands from users pertaining to displayed messages. For instance, users approval to continue or exit the application. Logging of following activities are emphasized by Error Handler. For a complete list of activities, see `SLCSLogger` class in appendix.

1. Key generation process. Logs the key length, algorithm used and name of the SE.

2. HTTP connection by application (e.g. to SLCS server).
3. Signing activity. Module requesting the signature generation, users response and success or failure of the activity.
4. The result of PIN entry. Logs module responsible for PIN entry and result.

A timestamp is always included with any logging activity.

Chapter 7

Analysis and Evaluation

7.1 Usage Scenarios Revisited

The usage scenarios described in Chapter 4 has served as basis for the requirements of the solution. In this Section we consider the usage scenarios from the point of view of how the proposed solution benefits them.

Mobile Traveler

The basic use case of mobile traveler involves bootstrapping short-lived certificate while inside the home network and using the certificate for authentication to services while traveling. The node-to-node interaction is extended with both parties authenticating each other.

Shibboleth based IdP as described in prototype implementation is increasingly being used for organizational identity needs. It is HTTP based and supports SAML message exchange to other parties such as SLCS. Mobile user, while inside the network, contacts first the organization's SLCS service. HTTP redirection from SLCS to Shibboleth IdP and subsequent authentication by IdP successfully provides a short-lived certificate in single session. SLCS client in mobile device uses CM's interface to ensure that credentials are secure in a SE chosen during SLCS application start-up. The short-lived certificate can be made available to service provider (or any party seeking authentication) using the SLCS client component of the solution. The solution provides basic PKI functions such as message signing and signature verification which can be used by application specific authentication algorithm through plug-in (see Section 6.4).

Our solution is generic in nature in terms of communication prototype between RA and mobile user. However, implementation tightly couples SLCS with Shibboleth . This results in restricting the organization to implement any other authentication architecture.

Single Sign-on

We discussed the Single Sign-on use case in Section 4.1. If security requirements are week, private keys can be stored in plain file without pass-phrase (or PIN) protection. The applications therefore can perform authentication using certificate and private keys without user's intervention.

The solution supports this feature using configuration parameters and SE. If the configuration allows plain file storage by setting the SE to plain file, CM component of the solution uses Java ME file connection API (JSR 75) for plain file storage. Since the abstraction is implemented in SE, other features provided by CM remains as it is. The applications performing single sign-on can interface with SLCS application in mobile device for location of private key and certificate. Indeed, all the services of CP and CM are available to other applications which can be used for example to request short-lived certificate through CP component of SLCS, to store and manage credentials through CM component or for performing digital signatures.

A major concern is concurrent support for plain file in device and temper resistant storage. As the solution requires SE set-up during SLCS initialization itself, only one storage element can be supported at a time. The application can be made multi-threaded with several parallel threads of CM each having its own configuration parameter. However, complexity increases as user (who is responsible for SE selection) needs to configure each SE individually and keep a track of it while using SLCS.

Mobile Signature Service

Signature framework as defined by ETSI are supported in the solution using temper resistant storage as security element. The CM component ensures that private keys never leave the card and cryptographic operations are performed in the card (using SATSA as specified in JSR 177) itself. The SLCS application does not pose any limitation on its own, rather it is the selection of SE (e.g. an external Smartcard) for storage or cryptographic operation that determines what level of signature support is available in our solution. For instance, if a smart card in the mobile device is capable to cary out 1024 bit RSA cryptographic operation, the CM component can simply forward the

request to SATSA API for actual operation in smart card. In this respect SE serves the purpose of SSCD as required in ETSI qualified signatures.

A possible contention raised by using SIM as SSCD for ETSI approved mobile signatures. Since SSCD has to be issued and approved by a CA, SIM manufacturers would not like to hand-over the control of manufacturer's key and data stored in the SIM. For cases, where SIM providers also serves as CA this issues can be settled. However, mobile users would always prefer not to bind their certificate providers with SIM providers.

7.2 Validation against Criteria

In this Section, we evaluate the proposed solution against the set of criteria listed in Chapter 4. We consider each criteria individually and focus on set of questions (`Q.n.n`) and matrices (`M.n.n`) in Goal Quality Matrix defined for each criteria (See Chapter 4 for details) . While analyzing criteria, we place the associated matrix label (`M.n.n`) where it most appropriately address a question in the criteria.

7.2.1 Usability Criteria

Criterion 1: The solution should be easy to use.

The solution provides menu based application with one point of entry and exit. Moreover, solution does not require installation of any third party software apart from standard Java ME packages (`M.1.1`).

Process of requesting and obtaining certificate by user is a central feature of the application. At each stage of activity (starting with menu driven option to generate a key pair) user is guided through a readable text on display about the activity being performed and next activity required (for details of stages involved in requesting and obtaining a certificate see Section 5.3. Each acticvity displays option to cancel or approve the acivity by user (`M.1.2`). Yet, every attempt has been made to hide the technical complexity from non technical users.

For the authentication process (and other activities where user permission is mandatory), the user is informed (with readable text including the certificate name and purpose of authentication) and approval is sought (using PIN) before proceeding (`M.1.3`).

This criterion can be considered fulfilled.

Criterion 2: The solution should be platform independent.

The solution uses SATSA-PKI [1] to communicate with SIM and other smart cards in mobile device. Although recently mobile phone manufacturers have started supporting SATSA (e.g. Nokia S60) not all mobile devices available in market support SATSA-PKI for SIM and smart card communications and continue to use their native applications for this purpose (`M.2.1`).

However, since Java ME is device independent development platform, opening the SIM/Smart card as per JSR 177 by manufacturer is only requirement for feasibility of solution to work with SIM/Smart cards (`M.2.2`).

From matrices `M.2.1` and `M.2.2`, we conclude:

This criterion can be considered partially fulfilled.

7.2.2 Technical Criteria

Criterion 3: Key pair (public and private keys) should be generated inside the card.

CM component of the solution is responsible for key pair generation. CM communicates using SATSA API to the smart card applet running inside the temper resistant card for key generation. Section 5.2 details the communication between device application (CM) and smart card which is based on APDU protocol.

We can conclude that the solution implements key pair generation inside temper resistant card (SIM or smart cards) through SATSA provides APDU communication between device application and SIM/Smart Cards (`M.3.1`).

This criterion can be considered fulfilled.

Criterion 4: Private keys should be securely stored.

In addition to key pair generation inside the card (see criteria 3), any operation such as signing text and digital signatures requiring private keys are performed inside the card (see 5.2. The requests are accepted in terms of text or bytes to be signed and only signed data is returned. Thus the solution ensures that private keys never leave the card (SIM or Smart Card)(`M.4.1`).

[1] SATSA extends the security features of standard Java ME platform through JSR 177. SATSA-PKI is one of the packages of the extension. See Section 3.3.4

Although, solution also provides private key store in plain text for week security needs which do not require card storage(see Section 5.2). However, this is not a requirement rather an additional feature to users. The solution ensures that if user chooses a plain text storage mode, a warning (that key storage is without any security) and associated description is clearly displayed to user.

This criterion can be considered fulfilled.

Criteria 5: Short-lived certificate request and issuance (to mobile client) should be online and real-time.

The solution requires that user's credentials are available with the user's IdP. This is necessary to avoid offline credential verification by the CA issuing certificate (see Section 6.2). The SLCS service which acts as RA, is a dedicated service listening over HTTP to which the CP connects for request and issuance.

The time consumed from the point a request is made to the point when certificate is stored in the mobile device can be divided into following parts. Time consumed by user to read details on screen and PIN entry, HTTP connection time and time spent in key generation in smart card. As table 7.1 shows, smart card timings are in order of milliseconds. Considering that user is required to read and approve the request (using PIN entry) which can take at least a couple of seconds, our experience with prototype implementation shows that the HTTP connection and (HTTP) request processing times are less than time spent by user. We can conclude that the average processing time for certificate request and issuance does not result in any long pauses (`M.5.1`).

This criterion can be considered fulfilled.

Criteria 6: Support for cryptographic operations using certificates

Java ME as a development platform for the application ensures that wide range of cryptographic algorithms as supported in in Java Cryptography Architecture Specification are available. In particular, Section 6.4 details the algorithms supported in prototype implementation (`M.6.1`). Also see Table 7.1, which shows an average of 3 different brands of smart card timing survey performed by Samuel.et.*al.* [16].

This criterion can be considered fulfilled.

Criterion 7: The solution should be scalable.

The solution poses no limits in terms of number of certificate than can be requested or

Table 7.1: Smart Card Operation Timings [16]

Operation	**Time (in sec)**
RSA Private Key Decryption (Key length = 1024 bits)	0.58
RSA Public Key Encryption (Key length = 1024 bits)	0.19
SHA-1 hash on 64 bytes of data	0.16
Triple-DES encryption in CBC mode on 64 bytes of data	0.28
Triple-DES encryption in ECB mode on 64 bytes of data	0.21
Random number generation (64 bytes)	0.43

number of key generations that can be performed. In fact these activities are bound only by storage limitations of SIM/Smart cards. Further, the strength of algorithm and key length is limited by cryptographic capabilities of card. The proposed solution does not impose any additional limitations (`M.7.1`).

This criterion can be considered fulfilled.

Criterion 8: Use of open standards in solution.

The solution uses standard X.509 certificates. The APDU protocol used for communication between device application and SIM/Smart Cards is an open standard. Besides, SATSA implementation is open standard with JSR 177 specifications (`M8.1`).

Also the SLCS server which is an external components of the solution, has been developed by EGGE Middleware as open source based on open standards. See table See table 7.2 for a list of standards associated with components of the solution.

We also notice that "open standards"(such as listed in the Table 7.2) may be limited in in terms of the "degree of standards" or how widely acceptable they are. For instance, SLCS server is an open source implementation and a standard in context of gLite Middleware. However, the specifications of SLCS (or usage) are not well known outside Grids and hence can not be termed as "standard" in general.

This criterion can be considered fulfilled.

Table 7.3 summerizes the valiadtion against criteria.

Table 7.2: The Solution Components and Standards Used

Component or Activity	**Standards Used**
SLCS Server	Open source implementation from EGEE Middleware (gLite)
Client Proxy	Standard HTTP package from Java ME
Credential Manager	Java ME standard packages, SATSA (JSR 177)
Security Element	Java Smart Card Simulator
Cryptographic operations	Java Cryptography Architecture Specification
Device-Card communication	APDU Protocol, implemented in JSR 177

Table 7.3: Summary of Validation against the Criteria

Criterion	**Description**	**Priority**	**Fulfilled**
1	Easy to use solution	1	Yes
2	Platform independent solution	2	Partially
3	Secure storage of private keys in device	1	Yes
4	Key pair generation inside the card	1	Yes
5	Online certificate request and issuance	1	Yes
6	Cryptographic operations support	1	Yes
7	Scalability of the solution	1	Yes
8	Use of open standards in the solution	2	Yes

Chapter 8

Conclusions

PKI is emerging as enabling security mechanism for accessing services over mobile phones. At the same time, mobile PKI brings its own set of challenges owning to resource constrained mobile device environment. Several established PKI mechanism for the Internet such as CRLs and offline credential verification for certificate issuance restrict the usage of PKI in mobile devices. Further, the portable nature of hand-held increases the vulnerability of private key misuse requiring secure private key store in hand-held devices.

In the thesis we introduced a PKI based mobile authentication method which addresses aforementioned challenges. We drew our motivation from the usage of short-lived certificates in EGEE Grids and proposed that our method could bring significant advantages in specific usages scenarios. We set the requirements of thesis work in context of usage scenarios namely mobile travelers and disconnected mobile users as we reasoned that they benefit most from the proposed method. User-friendliness, platform independency, use of open standards and scalability have been objectives of the solution and we laid down them as objectively as we could in terms of usability and technical requirements.

We demonstrated that the short-lived certificates can be used to ease the requirements of CRLs and it can facilitate authentication in disconnected scenarios where neither party is connected to any identity provider. We have also shown that short-lived certificates can be used for standard mobile signatures since the storage of private keys as well as cryptographic operations on them are performed inside temper resistant cards.

Evaluating criteria objectively is always a challenge. We adapted well researched GQM approach for criteria evaluation by listing a set of questions and associated matrices

for each criteria (as defined in requirement phase). We addressed each such question, provided answer in form of matrices, reasoned our answer by referring to the solution and evaluated to what extent the solution fulfills the criteria.

8.1 Directions for Future Work

An open concern in any PKI based authentication is technologies for secure storage of credentials. Mobile PKI further raises the challenge as mobile devices are easily accessible to public and highly vulnerable to misuse. Temper resistant storage with PIN protection has emerged as secure and convenient solution. However, there is a still a lack of agreement among mobile operators over Device vs. SIM/SmartCard communication standards. Even as industry wide standard such as APDU exists and SATSA implements it, manufacturers are unwilling to allow access of SIM card to third party applications. A research into device independent standards for communication between device applications and SIM/SmartCards as well as practical issues restricting manufacturers to adapt them could greatly leverage mobile commerce.

Anther open issue which restricts the usage of PKI in general and mobile PKI in particular is offline credential verification by RA. Not only it adds cost, the delay in the process reduces the usability of PKI based applications. Federated IdPs and bootstrapping credentials address this issue but they are highly dependent on organization's policy and implementation. A research into online RA that could request and handle credentials from IdPs based on different technologies (Shibboleth is one such technology) could dramatically improve the usability of PKI based security solutions.

Bibliography

[1] Cryptographic Message Syntax. RFC 2630. http://www.ietf.org/rfc/rfc2630.txt.

[2] Enhanced Security Services for S/MIME. RFC 2634. http://www.ietf.org/rfc/rfc2634.txt.

[3] Hypertext Transfer Protocol. RFC 2616. http://www.ietf.org/rfc/rfc2616.txt.

[4] Internet X.509 PKI and Certificate Revocation List (CRL) Profile. RFC 5280. http://www.ietf.org/rfc/rfc5280.txt.

[5] PKCS 10: Certification Request Syntax Specification. RFC 2968. http://www.ietf.org/rfc/rfc2986.txt.

[6] PKCS 7: Cryptographic Message Syntax. RFC 2315. http://www.ietf.org/rfc/rfc2315.txt.

[7] 3G Security: Generic Authentication Architecture (GAA); System description. TR 33.919, 3rd Generation Partnership Project (3GPP), 2004. http://www.3gpp.org/ftp/Specs/html-info/33919.htm.

[8] M-Shield mobile security technology. Technical report, Texas Instruments, 2005. http://focus.ti.com/pdfs/wtbu/ti mshield whitepaper.pdf.

[9] *Java Cryptography Architecture (JCA) Reference Guide*, 2006. Java™ Platform Standard Edition 6.

[10] T. Abe, H. Itoh, and K. Takahashi. Implementing identity provider on mobile phone. In *DIM '07: Proceedings of the 2007 ACM workshop on Digital identity management*, pages 46–52, New York, NY, USA, 2007. ACM.

[11] G. An, J. Kwak, and D. Won. Wireless authentication protocol keeping low computation in mobile environment. page 6. WSEAS, Sept. 14-17 2003.

[12] N. Asokan, V. Niemi, and P. Laitinen. On the Usefulness of Proof-of-Possession. In *Proceedings of the 2nd Annual PKI Research Workshop*, pages 122–127, 2003.

[13] V. Basili, G. Caldiera, and H. D. Rombach. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*, pages 528–532. John Wiley and Sons, 1994.

[14] I. Bate. Systematic approaches to understanding and evaluating design trade-offs. *Journal of Systems and Software*, 81(8):1253–1271, Aug. 2008.

[15] H. Brandl. Introduction To Trusted Computing: The TCG Trusted Platform Module Specification. Technical report, 2006.

[16] T.-W. Cheung and S. T. Chanson. Design and Implementation of a PKI-Based End-to-End Secure Infrastructure for Mobile E-Commerce. In *WISE (1)*, pages 3–7, 2001.

[17] C.Witzig. Shibboleth interoperability through a short-lived credential services. Technical report, SWITCH EGEE-II, 2006.

[18] J. Dankers, T. Garefalakis, R.Schaffelhofer, and T.Wright. Public Key Infrastructure in Mobile Systems. *Electronics and Communication Engineering Journal*, pages 180–190, Oct. 2002.

[19] J. Ekberg, N. Asokan, K. Kostiainen, and A. Rantala. OnBoard credentials platform design and implementation. Technical report, Nokia Research Center (NRC), 2008. NRC-TR-2008-007.

[20] European Telecommunications Standards Institute. *Mobile Commerce (M-COMM); Mobile Signature Service; Web Service Interface*, 2008. Technical Specifications, ETSI TS 102 204 V1.1.4 (2003-08).

[21] P. Gutmann. Simplifying Public Key Management, 2004.

[22] S. Hawkins, D. C. Yen, and D. C. Chou. Awareness and challenges of Internet security. *Information Management and Computer Security*, 8(3):131–143, 2000.

[23] C. Heath. *Symbian OS Platform Security*. John Wiley & Sons, 2006.

[24] J. Herstad, V. Thanh, Do, and S. Kristoffersen. Wireless Markup Language as a Framework for Interaction with Mobile Computing and Communication Devices. In *Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices*, page 13, Glasgow, Scotland, 1998-05-21/1998-05-22.

[25] Y.-K. Hsu and S. P. Seymour. An Intranet Security Framework Based on Short-Lived Certificates. *IEEE Internet Computing*, 2(2):73–79, 1998.

[26] K. Hyppönen. An Open Mobile Identity Tool: An Architecture for Mobile Identity Management. In *5th European PKI Workshop: Theory and Practice, EuroPKI 2008, June 16-17, 2008, Proceedings*, volume 5057 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2008.

[27] J. Iliadis, D. Spinellis, D. Gritzalis, B. Preneel, and S. Katsikas. Evaluating certificate status information mechanisms. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 1–8. ACM Press, Nov. 2000.

[28] M. Ishikawa and S. McCarron. Module-based XHTML -Second Edition. Working draft, W3C, Feb. 2007. http://www.w3.org/TR/2007/WD-xhtml11-20070216.

[29] Java Community Process. JSR 177: Security and Trust Services API for J2ME, 2004. http://jcp.org/aboutJava/communityprocess/final/jsr177/index.html.

[30] B. Kaplan and D. Duchon. Combining qualitative and quantitative methods information systems research: a case study. *Management Information Systems Quarterly*, 12(4):571–586, 1988.

[31] K. Kawamoto and N.Nakamura. Study of Management on the Mobile Public Key Infrastructure. NTT DoCoMo R&D Center, 2002.

[32] A. Kettula and A. Kettula. Security Comparison of Mobile OSes. In *Helsinki University of Technology*, 2000.

[33] T. Kwon. On the difficulty of protecting private keys in software. In *ISW: International Workshop on Information Security, LNCS*, 2002.

[34] P. Laitinen, P. Ginzboorg, N. Asokan, S. Holtmanns, and V. Niemi. Extending cellular authentication as a service. *The First IEE International Conference on Commercialising Technology and Innovation, 2005*, Sept 2005.

[35] Y. Lee, J. Lee, and J. Song. Design and implementation of wireless PKI technology suitable for mobile phone in mobile-commerce. *Computer Communications*, 30(4):893–903, 2007.

[36] R. Molva, D. Samfat, and G. Tsudik. Authentication of Mobile Users. *IEEE Network*, 8:26–34, 1994.

[37] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein. Federated Security: The Shibboleth Approach. *EDUCAUSE Quarterly*, 27(4):4–6, 2004.

[38] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public-Key Infrastructure: Online Certificate Status Protocol (OCSP). RFC 2560, June 1999. http://www.ietf.org/rfc/rfc2560.txt.

[39] Y. Nakajima, H. Masamoto, K. Kobayashi, M. Sakamoto, and S. Arai. CLIP: An Online PKI Management Function. *NTT Technical Review*, 6(3), 2007.

[40] C. E. Ortiz. The Security and Trust Services API for J2ME, March 2005. http://developers.sun.com/mobility/apis/articles/satsa1/.

[41] M. Pitkänen and H. Mikkonen. Initializing Mobile User's Identity From Federated Security Infrastructure. In *Proceedings of the Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM '08)*, pages 390–394, Valencia, Spain., Sept.–Oct. 2008.

[42] R. S. Pressman. *Software Engineering.* McGraw-Hill Inc., third edition, 1992.

[43] Rheinisch-westfalische, L. F. Informatik, S. Holtmanns, and P. Wesnarat. Identity Management in Liberty Alliance Project, 2004.

[44] A. Romao and M. M. da Silva. Secure Mobile Agent Digital Signatures with Proxy Certificates. In *Lecture Notes in Computer Science*, volume 2033/2001, pages 1–65. Springer Berlin / Heidelberg, Berlin, 2001.

[45] A. D. Rubin and D. E. Geer. Mobile code security. *IEEE Internet Computing*, 2(6):30–34, Nov.–Dec. 1998.

[46] A. Ruiz-Martínez, D. Sánchez-Martínez, M. Martínez-Montesinos, and A. F. Gómez-Skarmeta. A survey of electronic signature solutions in mobile devices. *Journal of Theoretical and Applied Electronics Commerce Research*, 2(3):94–109, 2007.

[47] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1994.

[48] M. Shaw. The Coming-of-Age of Software Architecture Research. In *23rd International Conference on Software Engineering (ICSE '01)*, pages 657–664a, May 2001.

[49] P. Tremblett. X.509 certificates. *Dr. Dobb's Journal of Software Tools*, 24(7):42, 44, 46, 47–51, July 1999.

[50] K. Vedder and F. Weikmann. Smart Cards – Requirements, Properties, and Applications. In B. Preneel and V. Rijmen, editors, *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography, LNCS 1528, 1998*. 1998.

[51] T. Weigold. Java-Based Wireless Identity Module. In *RFP, I-Logix, Open-IT, and THALES*, pages 03–08, 2002.

[52] M. Winslett, T. Yu, K. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating trust in the Web. *Internet Computing, IEEE*, 6(6):30–37, Nov/Dec 2002.