NTNU
Norwegian University of Science and Technology

KTH VETENSKAP OCH KONST
ROYAL INSTITUTE OF TECHNOLOGY

MASTER'S THESIS

A secure mobile phone-based interactive logon in Windows

Oleksandr Bodriagov

Master of Science in Security and Mobile Computing
Submission date: June 2010
Supervisor - Ubisafe AS: Ivar Jørstad
Professor - NTNU: Do van Thanh
Associate Professor - KTH: Karl-Johan Grinnemo

Norwegian University of Science and Technology
Department of Telematics

The Royal Institute of Technology
School of Information and Communication Technology

Problem description

The goal of this thesis is to study and propose how the Windows identity management can be made more secure and user-friendly by using a mobile phone-based workstation logon scheme. The current logon process is neither sufficiently secure nor user friendly.

The thesis also investigates how new authentication schemes in general and those that work with mobile phones in particular could be integrated into the Windows logon system. It analyses the complexity of integrating new authentication solutions into Windows and describes what Windows components need to be customized/modified in order to incorporate a new authentication method into the logon procedure.

This work consists of the following tasks:

- A study of existing identity management and authentication standards, protocols, and solutions for enterprise environments.
- A study of the Windows platform security architecture
- An analysis of Windows extendibility: custom authentication mechanisms
- A study of existing mobile phone-based authentication schemes
- Design and implementation of a solution for the mobile phone-based workstation logon process in the Windows operating system
- Security analysis of the proposed solution

Assignment given: 2010-01-15

Supervisor: Dr. Ivar Jørstad, Ubisafe AS

Preface

The thesis is submitted to the Norwegian University of Science and Technology (NTNU) and to the Royal Institute of Technology (KTH) as a final work of the joint Master's Programme in Security and Mobile Computing (NordSecMob) to fulfill requirements for the Master of Science degree.

This master's thesis has been supervised by the Dr. Ivar Jørstad from the Ubisafe AS, while the academic responsible have been Professor Do van Thanh from NTNU and Associate Professor Karl-Johan Grinnemo from KTH. I would like to thank them all for the provided help, suggestions, and guidelines during my work on the thesis, especially to Ivar Jørstad and Do van Thanh.

Trondheim, June 2010.

Oleksandr Bodriagov

Abstract

Password-based logon schemes have many security weaknesses. Smart card and biometric based authentication solutions are available as a replacement for standard password-based schemes for security sensitive environments. However, the cost of deployment and maintenance of these systems is quite high. On the other hand, mobile network operators have a huge base of deployed smart cards that can be reused to provide authentication in other areas significantly reducing costs.

This master's thesis presents a study of how the workstation identity management can be made more secure and user-friendly by using a mobile phone in the Windows workstation logon process. Two workstation logon schemes that utilize both the mobile phone and the UICC inside of the phone are proposed as a result of this study.

The first scheme emulates a smart card reader and a smart card in order to interoperate with the Windows smart card framework to provide PKI-based logon. The mobile phone with the UICC card emulates a smart card that communicates with the emulated smart card reader via protected Bluetooth channel.

The proposed scheme reuses the Windows smart card infrastructure as much as possible, both in terms of software and hardware. Therefore, a seamless integration with Active Directory and Window server is achieved. This scheme can work with any authentication scheme used with real smart cards. It can be used not only for the logon but also for all other functions typically done with smart cards (e.g. signing of documents, e-mails).

In the second scheme, the mobile phone with the UICC serves as a token for generating OTP values based on a shared secret key and the time parameter.

In order to design Windows logon architectures based on mobile phones, a study of relevant technologies, components, and their security aspects has been conducted. Existing phone-based authentication schemes have been thoroughly studied both from the usability and from the security points of view. This has been done to understand possible alternatives for different aspects of the architectures that were designed.

The thesis analyzed how new authentication schemes in general and those that work with mobile phones in particular could be integrated into the Windows logon system. A conclusion is made that it is impossible to make a generic architecture that would easily support all existing and possible future mobile phone authentication schemes for the Windows logon. Windows is already a highly customizable environment and can support virtually any authentication scheme for the logon, though a considerable amount of modifications may be required to implement a particular scheme.

Table of Contents

List of figures

List of tables

Abbreviations

3GPP	The 3rd Generation Partnership Project
AID	Application identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
AS	Authentication Server
ATR	Answer-To-Reset
AuC	Authentication Centre
AUTN	Authentication Token
BC	Broadcast domain
CAPI	Cryptography API
CBC-MAC	Cipher Block Chaining Message Authentication Code
CID	Channel Identifier
CK	Encryption key
CNG	Cryptography API: Next Generation
CS	Circuit Switched
CSP	Cryptographic Service Provider
DAC	Discretionary Access Control
DACL	Discretionary Access Control List
DLL	Dynamic-link Library
DPA	Differential Power Analysis
EAL	Evaluation Assurance Level
EAP	Extensible Authentication Protocol
EAP-AKA	EAP Method for 3rd Generation Authentication and Key Agreement
EAP-SIM	EAP Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
EEPROM	Electrically Erasable Programmable Read-Only Memory
EF	Elementary File
EIR	Equipment Identity Register
FAST	Flexible Authentication Secure Tunneling Protocol
GAP	Generic Access Profile
GGSN	Gateway GPRS Support Node
GINA	Graphical Identification and Authentication
GMSC	Gateway Mobile Switching Center
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HLR	Home Location Register
HOTP	HMAC-Based One-Time Password Algorithm
HSS	Home Subscriber Server
IAM	Identity and Access Control Management
IDM	Identity Management
IDMP	Identity Device with Microsoft Generic Profile
IK	Integrity Key
IMEI	International Mobile Equipment Identity
IMPI	IP Multimedia Private Identity
IMPU	IMS Public Identity
IMS	IP Multimedia Subsystem

IMSI	International Mobile Subscriber Identity
IOCTL	I/O Control
ISIM	IP Multimedia Services Identity Module
ISM	Industrial, Scientific and Medical
J2ME	Java 2 Platform, Micro Edition
KDC	Key Distribution Center
KSP	Key Storage Provider
L2CAP	Logical Link Control and Adaptation protocol
LDAP	Lightweight Directory Access Protocol
LFSR	Linear Feedback Shift Register
LMP	Link Manager Protocol
Logon UI	Logon User Interface
LSA	Local Security Authority
MAC	Message Authentication Code
ME	Mobile Equipment
MNO	Mobile Network Operator
MSC	Mobile Switching Center
MSISDN	Mobile Subscriber ISDN Number
NIS	Network Information Service
OBEX	Object Exchange
OS	Operating System
OTA	Over-The-Air
OTP	One-time Password
PC/SC	Personal computer/Smart Card
PDU	Protocol Data Unit
PIN	Personal Identification Number
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
PNP	Plug and Play
PS	Packet Switched
P-TMSI	Packet-Temporary Mobile Subscriber Identity
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RAND	Random Number
RFCOMM	Radio Frequency Communication
ROM	Read Only Memory
SAML	Security Assertion Markup Language
SAS	Secure Attention Sequence (CTRL+ALT+DEL)
SASL	Simple Authentication and Security Layer
SATSA	Security and Trust Services API
SDP	Service Discovery Protocol
SGSN	Serving GPRS Support Node
SID	Security Identifier
SIM	Subscriber Identity Module
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSO	Single Sign-On
SSP	Secure Simple Pairing

TGS	Ticket Granting Service
TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity
TOTP	Time-based One-time Password Algorithm
UE	User Equipment
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications System
USAT	USIM Application Toolkit
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module
UUID	Universally Unique Identifier
VLR	Visitor Location Register
XML	Extensible Markup Language
XRES	Expected Response

1. Introduction

Identity management is one of important concepts of the modern society. Every person has one or several identities. An ordinary person typically has a citizen identity, an employee/student identity, an alumni identity, a driver license, and a set of digital identities. People constantly need to prove their identity. While in the physical world this usually means showing a document that identifies you, in the digital world this process can be much more sophisticated. Digital identity attributes such as login, name, etc. can be easily copied, thus to prevent the identity theft some secret credentials known only to a person and an authentication authority have to be used [1].

The identity management is a continuous process that encompasses among other things identity lifecycle management, authentication, and access control [4]. In the context of modern enterprise information systems, the main purpose of the identity management is to manage access to enterprise resources and information assets. Enterprise information systems provide variety of services to support business processes, and with development of e-business the complexity and functionality of these systems only grows. Thus, it is becoming increasingly harder to provide a full protection from unauthorized access. Authentication plays a crucial role, since a decision whether to give access to services/resources is identity-based. Therefore, it is important to ensure that a fraudster would not be able to steal the identity or impersonate a user to a system in some other way.

1.1 Motivation

Authentication is done by presenting a proof of identity to the verifier. All authentication schemes are based on the combination of the following factors: something you know, something you have, and something that you are. The most common authentication scheme nowadays is a static password authentication (one factor authentication). Password authentication schemes have been used for a very long time. These schemes are easy to use, and people are used to them. However, from a security point of view these schemes have many weaknesses. The computational power of modern computers is constantly increasing, thus it is possible to launch a brute force extensive search attack on password-based authentication schemes if passwords are weak. Besides, new vulnerabilities that dramatically decrease the theoretical security level of cryptographic protocols are constantly found, thus allowing attackers to launch much more efficient attacks than the extensive search.

In a properly designed cryptographic scheme, the use of longer keys that have higher entropy provides a higher security level. Therefore, it might seem that by using strong, randomly generated passwords with adequate length that consist of a combination of letters, numbers, and special symbols we can solve this problem. However, such passwords are difficult to remember for humans. That is why people tend to choose passwords that can be compromised with a simple dictionary attack. Furthermore, the number of user-name/password systems that the user has to use can be quite big. Therefore, it can be difficult to remember all credentials. Consequently, some users either reuse passwords or write them down. It is also important to note that by using strict administrative measures one may not reach the aim of strengthening security of a system. A too strict password policy (hard to remember passwords, frequent change, etc.) instead of strengthening the overall security of the system can actually weaken it, since users can end up writing down passwords. The usage of default passwords and careless users, which reveal their passwords either accidentally or as a result of social engineering attacks, can further reduce the security level of the system.

For some systems, the password-based authentication may be sufficient, but systems that process sensitive data require stronger authentication schemes. Biometric, smart card based, or one-time password based authentication schemes are considered to be much stronger

than the ordinary user-name/password authentication scheme, though the cost of deployment and maintenance makes these systems less common.

Security is a cornerstone in a smart card development. A combination of logical and physical security mechanisms, which form a unified protection system, ensures a high level of security. The ability to store information (e.g. identity information) and execute cryptographic protocols in a secure manner resulted in a great success for smart cards in security sensitive areas. Mobile network operators have already deployed smart cards to authenticate subscribers, thus this infrastructure can be reused to provide authentication in other spheres.

An authentication solution that uses a mobile phone with a UICC card as a security token can provide strong two-factor authentication based on the possession of the UICC card and on the authentication of the card owner to the card. This authentication solution provides much stronger level of security than the user-name/password authentication and at the same time reduces operational costs [99]. One of the most common areas of password authentication schemes usage is workstation/domain logon. Authentication that utilizes mobile phones can be a cost-effective and secure solution for the workstation logon in Windows domains.

1.2 Problem definition

The problem statement of this master's thesis is: "*How can the workstation identity management be made more secure and user-friendly by using a mobile phone in the Windows workstation logon process?"*

The master's thesis also investigates how new secure authentication technologies that utilize mobile phones as security elements can be integrated into the Windows environment. It analyses the complexity of integrating new authentication solutions into Windows and describes what Windows components need to be customized/modified in order to incorporate a new authentication method into the logon procedure.

1.3 Objectives

The main objective of the master's thesis is to develop a solution for the Windows workstation logon that would provide stronger security than the ordinary password scheme by using abilities of the phone itself and the card inside. The solution should not just concentrate on some particular phone. It should be compatible with as many phones as possible.

The second objective is to study how new authentication schemes that work with mobile phones can be integrated into the Windows logon system.

The task involves a study of related technologies and components and their security aspects. The study of identity and access management technologies, smart card related technologies, UMTS systems, and Bluetooth technology is conducted.

Existing mobile phone based authentication schemes are thoroughly studied both from the usability and from the security points of view. This is done to understand possible alternatives for different aspects of the architecture that is to be designed.

The next step is to design and implement the proposed solution. Then goes a discussion of the solution's security aspects and a comparison of the proposed solution with the strongest existing mobile phone based authentication schemes in terms of security, usability, and complexity of integration into Windows.

1.4 Related work

A lot of research concerning various aspects of the mobile phone based authentication has been done and several solutions have been proposed [99, 100, 103]. Most of these solutions are mobile network operator dependent. It means that the operator provides identity

management services.

The Mobile-OTP is a free one-time password authentication solution that works as a Java-based soft token system [111]. It means that a mobile phone with a Java MIDlet acts as a token that generates one-time passwords, and the UICC card capabilities are not used at all. This system was designed to be used for authentication of users at routers, firewalls, web servers, access points, and UNIX machines.

The ActivIdentity Corporation in April 2010 introduced a new solution to provision public key infrastructure (PKI) onto BlackBerry smart phones using a secure microSD card [112]. Their solution allows signing emails and provides a two-factor authentication for web portals. This solution is more secure than the previous one, but still not as strong as the one based on smart cards. Besides, it does not work with the Windows workstation logon.

Mobile phone based authentication schemes have been used in online banking and e-commerce for several years. In May 2010 EZMCOM launched MSIGN - Mobile PKI Platform [113]. It offers two-channel, two-factor authentication based on PKI. It utilizes an over-the-air (OTA) channel and a Java software client on the phone. This platform provides secure online banking, mobile payments, and e-commerce. Although this system provides strong two-factor authentication based on PKI enabled mobile phones, in its current form it cannot be used for the workstation logon.

To my knowledge, there is no available system or a published work that deals with Windows workstation logon schemes based on a mobile phone and a smart card inside of it.

1.5 Organization of the thesis

The thesis is organized as follows. The second chapter provides a study of relevant technologies. It contains an overview of what is identity management and a study of identity management in enterprises and UMTS systems. The chapter continues with an in-depth analysis of smart security and Bluetooth technology and security.

The third chapter “Analysis” investigates authentication and authorization in Windows. The emphasis on the logon procedure and the Windows smart card architecture is made. Then an evaluation of Windows extendibility in relation to custom authentication mechanisms is conducted. The chapter ends with a description and a thorough analysis of existing mobile phone based authentication schemes.

The fourth chapter contains two proposed logon architectures. The first one utilizes PKI and a Bluetooth smart card reader driver. The second one is a typical OTP scheme.

Chapter 5 is devoted to discussions and future work. Conclusions are made in chapter six. The document also has two appendixes that contain the source code for the proposed solutions.

2. Background and Overview

This chapter provides a study of relevant technologies. It is crucial for understanding of the thesis, since in this chapter many security aspects of the involved technologies are discussed.

2.1 Identity and identity management

This subchapter contains an overview of what is identity management and a study of identity management in enterprises and UMTS systems.

2.1.1 Notion of identity and identity management

Identity is an intuitively understood notion that is related to existence of objects, their properties and distinct characteristics [1]. It is used to distinguish objects. The classical view of identity is an equivalence relation defined as "everything has to itself and to nothing else" that complies with the Leibniz's Law [2]. The Leibniz Law that can be expressed as: "No two objects have exactly the same properties" consist of two principles [1]:

- If two objects are identical, then they must have the same properties
- If two objects have all the same properties, they are identical

Real-world objects have infinite number properties. Therefore, the second principle should be limited to a finite set of properties that can uniquely characterize an object among a specific group of objects in order to be used in the real life. This set of properties that uniquely identifies an object can be seen as an identity.

An identity is a set of attributes and identifiers associated with an entity [1] by which an entity is uniquely recognizable [3]. The following three definitions describe the actual purpose of the identity concept:

- Identification – is a process of determining the identity of an entity based on the identifier presented by the entity;
- Authentication – a process of verifying the identity of an entity [3];
- Authorization – the process of specifying access rights to resources of a system.

However, in the digital world it is relatively simple to make an exact copy of all attributes and identifiers. Therefore, identities used in digital world contain a secret attribute known only to the owner of the identity and, possibly, to an entity that performs authentication (in case of shared secret key authentication).

Identity management

The diversity of information systems and applications that a typical user has to work with is quite big nowadays. If these systems/applications are for the personal use, then it is responsibility of the user to manage all his/her identities among all these systems, which are quite often completely unrelated an independent from each other. Some users experience difficulties managing their identities because of the number of the involved systems. In managed environments with a big number of users, this problem is much more serious. The managing authority should ensure that all users receive access to data and applications that they are supposed to and, at the same time, it should ensure that access is not provided to any other subjects, thus protecting valuable assets. Since most of the access decisions are identity-based, identity management plays a fundamental role in the security of a system.

Identity management (Identity and access management) can be defined as a set of "*processes, policies and technologies to manage the complete life cycle of user identities across*

the system and to control the user access to the system resources by associating user rights and restrictions" [4]. A comprehensive identity management system includes the following functions [4]:

- Identification: defining user identity
- Authentication: verification of user identities
- Authorization: only authorized users can access protected resources
- Single Sign-on: allows a user to sign once in the system and have access to all required resources automatically without the need to provide credentials repeatedly
- Administration: managing users and assets. Includes registration, self-service, delegated administration, federated provisioning, etc.
- Auditing: documenting events and system activity

2.1.2 Identity and access management in enterprise systems

The main purpose of enterprise information systems is to provide a variety of services to support business processes. Business objectives are the driving force for the growth of functionality provided by IT systems. And with the development of e-business, enterprise IT systems get more and more tied to the Internet. However, the complexity and the vast functionality of these systems makes it increasingly harder to provide protection from unauthorized access to enterprise information assets. Access control management needs to take into account threats from outsiders, semi-trusted parties like partners, customers, and insiders. The access control system strongly relies on the identity management system to manage access to enterprise resources. Identity and access control management (IAM) must keep up with the demand of business to have external connections to partners, suppliers, customers, etc. Therefore, it is not enough to manage only identities of corporate users; identity management goes beyond the borders of an enterprise.

Situation is further complicated by the fact that many organizations have many heterogeneous IT platforms, each with its own identity database. The centralization of identity management simplifies management tasks, reduces management costs, and mitigates the risk that something will not be taken into account during data flows planning and privileges setup [5].

Moreover, IAM systems must comply with specific legislation if companies want to provide specific services like processing credit card payments [5].

Centralized identity and access control management

A centralized identity and access control management approach provides a single interface to manage identities and access controls for many heterogeneous systems with different repositories within an enterprise. A directory service forms a foundation of the identity and access management system because it stores information about a state of enterprise systems. This state information includes identities, account-related information (ex. passwords), policies, groups, roles, workflows, etc. A directory is a special type of database that is optimized for the read operation, while databases are optimized for the write/modify operation [9]. Besides, directories are mainly accessed via the Lightweight Directory Access Protocol (LDAP) and databases use the Structured Query Language (SQL) for this purpose [9]. If there is only one server that stores all data in one location, then the directory is centralized. The directory can also be distributed, when there is more than one server and information is either replicated

between servers, so that all have the same data, or it is dived between servers so that each holds only some part of data.

Many different repositories make it difficult to manage identities and access control rules, and this approach is also prone to mistakes that can lead to flaws in security. Synchronization and consolidation of data helps to reduce management costs and makes the whole system more manageable. There are many different ways in which an integrated enterprise directory service can be provided. It can be a single directory, a meta directory, and a virtual directory [5].

Single directory: as a name implies there is only one directory that is a single source of identity information for the whole system. A single directory simplifies management tasks compared to a bunch of different repositories; however, some applications might have to be modified to be able to work with a single directory [5]. Uniqueness is not in physical terms but in logical. A single directory can be distributed among several servers (each holding the same data) in order not to create a single point of failure and to distribute the load. Legal, political or security issues may make a barrier preventing creation of a single directory [10]. In this case, IAM provides a single interface to manage all these systems.

Meta directory: information from all various repositories is copied into a single directory with a unified namespace. Bidirectional synchronization mechanism controls synchronization between a meta directory and satellite/original repositories when data change occurs [10]. With this approach, there is no need to modify applications that worked with particular repositories since these repositories remain in the system [5].

Virtual directory: virtual directory serves as an abstraction layer between various repositories and applications by providing a single logical directory that gathers information from all repositories in real-time. A logical presentation of data can be customized for each application. From the same input data, different application-specific views of data, optimized for application needs, are derived [26]. There is no central physical directory that contains a copy of all data like in the meta directory approach.

IAM main functions:

- Managing users and accounts: includes creating, editing, deactivating, and deleting users, setting/changing user passwords, etc.
- Policy- and workflow-based management: Policies and workflows help automate management process. Policies that can be used in an enterprise include the following: account provisioning, password, authentication policy, etc. A workflow is a predefined sequence of automated processes that automates some time-consuming actions like gathering approvals. Workflows enforce consistency (the sequence and the set of involved actions is constant) and completeness (do not start some action until all previous are finished successfully). Request for approvals always take the same predetermined path and are automatically delivered to a person in charge.
- Privileges and Access control management: setting permissions so that only those entities that have permission can access a resource. The common models for access control are: the Discretionary Access Control (DAC), the Mandatory Access Control (MAC), and the Role-based access control model.
- Enterprise single sign-on: Single sign-on is a mechanism that enables a user to authenticate once to a system and reuse this authentication for many enterprise applications during current session. There is no need to have different accounts for different service providers and remember corresponding passwords. A user gets enterprise-wide access to data.
- Strong authentication: Strong authentication solutions imply the usage of a two factor authentication. Technologies such as smartcards and biometrics can be used

to achieve two-factor authentication. According to [10], SIM-based strong authentication that utilizes mobile phones can be quite cost-efficient compared to other two-factor solutions, and is comparable in price with password-based authentication.

- Monitoring and audit: system events are automatically checked against policies and regulations for violations. If the violation is detected, some action is triggered, for example account blocking.
- Federated identity management: user authenticated identity information is communicated across security domains to trusted partners that reside in the same Circle of trust. It means that a user does not need to authenticate when he wants to access resources of a collaborating company.

Identity and access manager architecture:

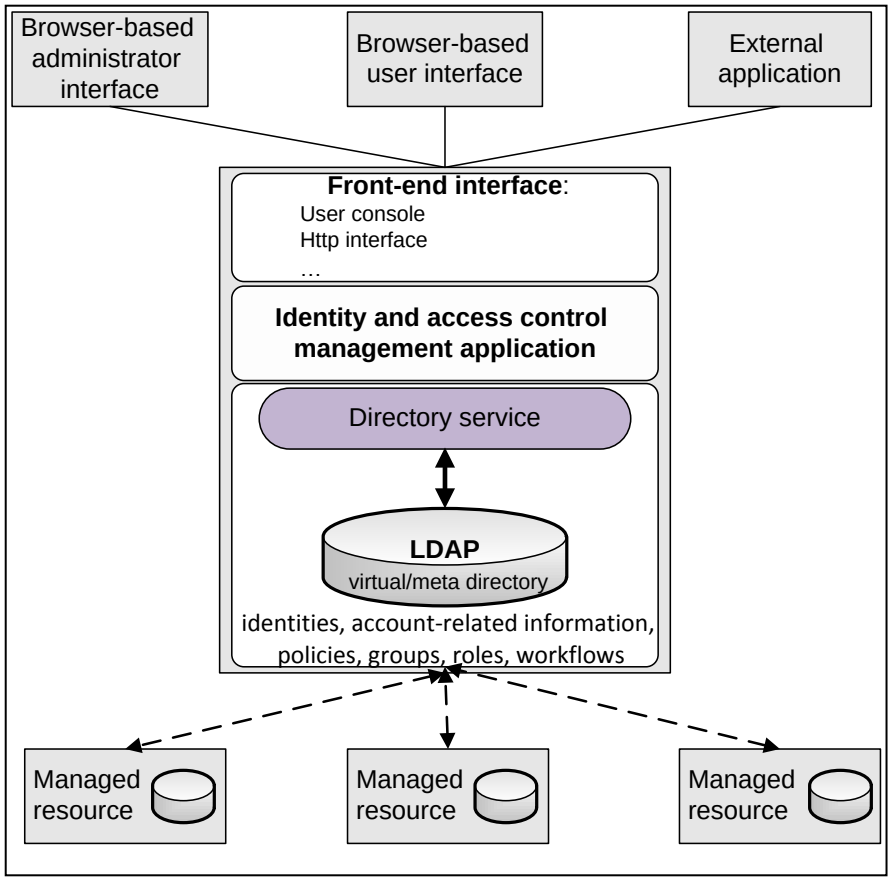

Figure 2.1.2.1: Identity and access manager architecture

It is possible to use a standard administration and management toolkit for a single directory without using IAM software. However, then all functions that are automated in an IAM suite should be done manually. IAM suites have a benefit that they can define policies that automatically create user accounts, mail boxes, and group memberships in real-time. Besides, it is not always possible to create a single directory.

Centralized authentication and trust

At the early stages of the development of authentication solutions a decentralized autonomous approach was used. Every station in a system performed authentication and authorization autonomously by maintaining its own users file/users database, and the access to services/resources provided by that station was based on that file. In order to allow users from one station to use services provided by some server, administrator had to add entries about those users to the users file of the server. A user had to have account on every station which

services/resources he used. This management approach had severe scalability issues. Remembering different passwords for different accounts, in case different passwords were used, or synchronizing password changes between many accounts, in case one password was used for many accounts - these both approaches introduced a lot of management and security problems. All identity and access control management procedures required enormous amount of work and were prone to errors.

The solution was to use a trusted third-party authentication approach. The third-party authentication authority maintains a centralized repository with identities and account-related information. Each user has only one account which is maintained by a third-party authority. The authentication is solely done by the trusted third-party and all other systems trust it. To be trustworthy the third-party should be highly secure, since the compromise of the third-party would lead to the compromise of a whole system. Besides, a third-party authority is a single point of failure, if it crashes the whole system will be inaccessible. There are two third-party authentication based schemas [12]:

- Implicit authentication schema: an authenticating entity (ex. Service provider) does not explicitly request authentication service from a centralized authentication authority. The authentication is cryptographically deduced from the encrypted message given by the third-party to the entity that is being authenticated. For example, Kerberos v5 protocol uses this approach.
- Explicit authentication schema: an authenticating entity explicitly requests third-party to make authentication.

There are many technologies that provide authentication and access control, but in general preferred are Kerberos security service and LDAP directories [13]. Kerberos primary provides authentication, however it can also provide some rudimentary authorization services. On the other hand, LDAP directories are mainly used for storing and managing authorization data, but they can also offer some authentication services [13]. That is why these technologies can be used separately or they can be integrated in one system.

Kerberos protocol

Kerberos is a protocol that deploys implicit authentication schema. Kerberos version 5, defined in RFC 4120, is the current version of the protocol. Kerberos performs mutual secure authentication in an unsecure network, though it does not provide accounting and can offer only very basic authorization [14]. Kerberos uses secret key cryptography to provide secure authentication service. There are three types of entities in the Kerberos architecture:

- Clients: want to use services provided by service providers
- Service providers: provide services for clients
- Kerberos servers: manage Kerberos authentication. They are called Key Distribution Centers (KDC). KDC shares a secret key with every principal (a client or a service provider) in a network. Clients and service providers trust KDC. KDC consist of three different elements: Authentication server that answers client authentication requests, Ticket granting server that issues Ticket Granting Service (TGS) tickets to clients, and a database that stores identities, secret keys, policies, etc.

Security in Kerberos schema is based on tickets and corresponding authenticators. A ticket is an encrypted message containing a client name, session key, and ticket's lifetime. The ticket is encrypted with a secret key shared by a server and KDC. The authenticator is an encrypted client's name, client's realm, timestamp. The authenticator is encrypted with the session key that is in the corresponding ticket. By providing the ticket and the authenticator to a server, a client can authenticate.

KDC is trusted by all principals in a realm - an authentication domain with some administrative boundaries. A realm is managed by only one KDC (KDC can be distributed

between several servers to share the load, but logically it is one). Each realm has its own KDC database. The principal identity consists of the Name and the Realm parts.

Kerberos authentication model

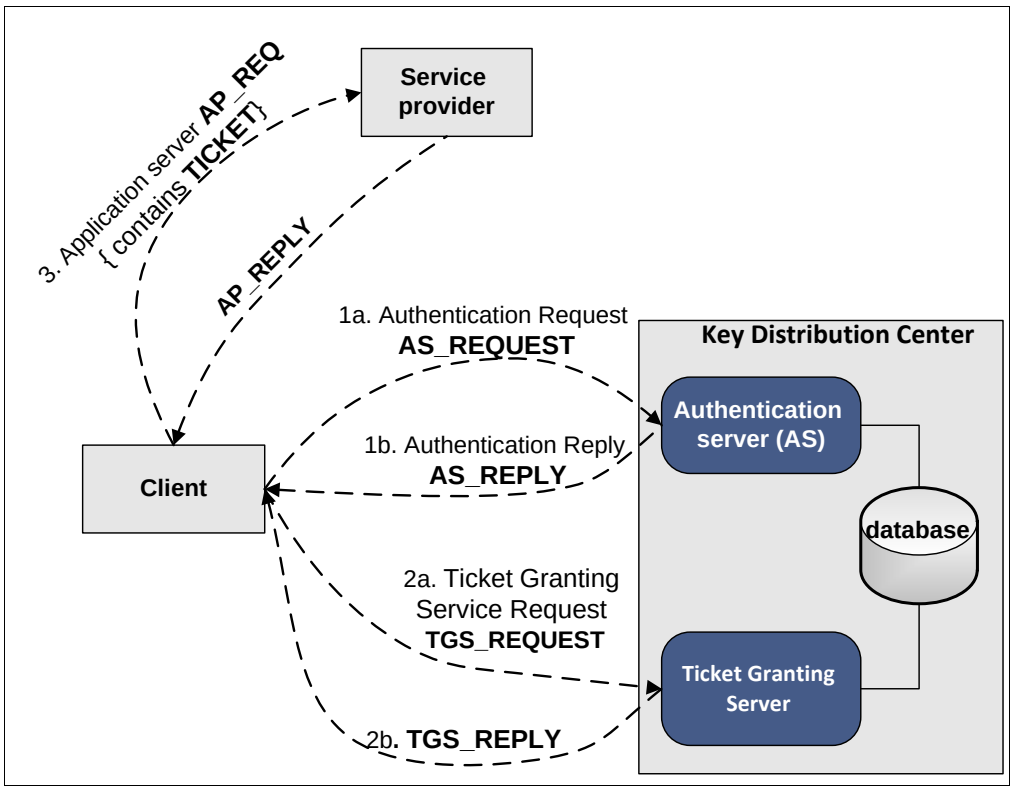

Figure 2.1.2.2: Kerberos authentication model

The whole Kerberos authentication procedure consists of three distinct exchanges:

- **The Client – Authentication server exchange**: The Client authenticates the KDC and obtains Ticket Granting Ticket (TGT) that will be used to obtain credentials for authentication to a Service provider. The Authentication server can authenticate the client before issuing the TGT (pre-authentication), or wait until Client – Ticket granting server exchange [13]. From the security point of view it is better to make pre-authentication, because this way the protection is provided against attacks on the principal's secret key [11], used by KDC to encrypt the response.
- **The Client - Ticket granting server exchange**: the Server authenticates the Client (this authentication happens in any case) and grants a Ticket Granting Service (TGS) to a Service provider specified in the request by the client. TGS will be used in the Client – Service provider authentication.
- **The Client – Service provider authentication exchange**: The Client sends a request to the Service provider which typically contains authentication information and initial request [11]. The Client always authenticates to the Service provider, but the mutual authentication must be requested by the Client explicitly [11]. The TGS and the authenticator which are in the Client's request enable the Service provider to authenticate the Client and derive shared session key which can be used to protect further communication with the client [11]. If the Client requests mutual authentication, then the Service provider's reply contains authentication information that enables the Client to authenticate the Service provider.

The main services provided by Kerberos are:

- mutual Client –Server authentication: a client is able to authenticate both KDC and various Service providers, so a malicious server won't be able to deceive a client and obtain confidential information.
- centralized management of authentication information: Kerberos utilizes centralized database to store identity and account information
- delegation: a principal may need to allow a service provider to perform operations on its behalf, for example a client can delegate rights to a printing server to access client's files on a file server to print them out [11]. A principal can ask KDC for a new TGT with a different network address, so that a service with that network address can act on behalf of the principal. The principal needs to transmit the new TGT and the corresponding session key to the service provider to enable delegation.
- single sign–on: Kerberos supports single sign-on by caching tickets and corresponding session keys. So the next time the authentication is needed, a user doesn't need to type in the password, cached credentials are used on user's behalf.
- cross-realm authentication: a client in one realm can authenticate to a service provider from another realm. To establish an inter-realm communication the service provider's KDC should be registered as a principal in the client's KDC. Then the client asks its KDC for a ticket to the service provider's KDC, presents this ticket to the service provider's KDC and asks it for a ticket to the service provider. It is possible to traverse several realms to authenticate to the remote service provider. Inter-realm communication can be organized hierarchically. This way the authentication path through multiple realms can be easily created [11].
- multi-factor authentication: the traditional Kerberos authentication is based on the secret keys (for a user the secret key is derived from a password). However, the public key cryptography for initial client – KDC authentication (PKINIT defined in RFC 4556) can also be used. Thus, it is possible to use smart cards and other cryptographic tokens when authenticating via Kerberos [17]. PKINIT requires the usage of trusted by KDC and its principals Certification Authority. It is also possible to use One-Time Password mechanism or biometric scanners for initial authentication.

There are several different implementations of the Kerberos Protocol: MIT version, Heimdal version, Windows Active Directory version, etc [15]. Active Directory not simply utilizes Kerberos as its default authentication protocol leaving NTLM for compatibility [7], but tightly integrates it in its framework, and that results in some issues in environments with Windows and non-windows systems [17].

Explicit authentication schema

The authentication is entrusted to a third-party that has all identity information. The typical authentication procedure is the following:

- a client contacts an application server and passes its credentials (identity and password) to the application server
- the application server contacts the trusted third-party server to perform the authentication

The exchanges between the client and the application server and between the application server and the third party should be cryptographically protected.

Technologies like Network Information Service (NIS), NIS+, and directory services with Lightweight Directory Access Protocol (LDAP) interface are used to provide centralized third-party directory service for storing identities and account-related information. Typically NIS is used to centralize the storage of /etc/passwd, /etc/shadow, /etc/group, etc. files from all

stations in the unix-based domain, and then this information can be accessed by clients [6]. The changes made in the centralized directory are propagated to all source stations. The NIS is an easy to manage protocol from administration point of view, but it does not provide security mechanisms: it does not provide authentication and authorization for directory access, and all communication is unencrypted [14].

Both NIS and NIS+ are Remote Procedure Calls (RPC) based protocols. NIS+ is a successor of the NIS protocol. In addition to providing hierarchical namespace, NIS+ offers a stronger security. However, NIS and NIS+ are legacy technologies, and it is recommended to migrate to LDAP [16].

Authentication against LDAP server

Lightweight Directory Access Protocol, defined in RFC 4511, is a protocol for accessing directory services that comply with X.500 standard. There are many directory servers that provide LDAP support: Active Directory, Novell eDirectory, Sun Java System Directory Server, IBM Tivoli Directory Server, OpenLDAP, etc.

LDAP is a client-server paradigm protocol that runs directly on top of TCP. It is recommended that the servers listen on port 389 for incoming requests. A client transmits a request to a server to perform some operation in the directory. The server performs the operation and returns a response. Some of the operations used in LDAP are the following: bind, unbind, search, modify, add, delete, compare, startTLS.

The Bind operation is equivalent to authentication. The Bind request specifies the authentication identity. The Bind operation utilizes several authentication methods: simple authentication method and SASL authentication method [25]. The simple authentication method is further divided into: anonymous authentication, unauthenticated authentication, and name/password authentication. In the name/password authentication the client sends both the name and the password to the server for validation. This authentication method should only be used in environments where confidentiality protection is provided [25]. Client can request TLS establishment for the LDAP session by sending request with StartTLS operation. TLS provides confidentiality and integrity protection for LDAP session, so it is possible to perform a simple name/password authentication in a secure manner.

Simple Authentication and Security Layer (SASL), defined in RFC 4422, is a framework that enables usage of various security mechanisms in protocols. SASL provides the abstraction layer that allows any protocol to utilize any mechanism. By providing SASL authentication method, LDAP allows authentication via any SASL mechanism [25].

The Bind response message is just an indication of a success/failure of the authentication request.

LDAP became one of the major elements in enterprise identity and access control systems [15]. It provides centralized storage for identity and account-related information and can be used to authenticate principals. The authentication procedure starts with client sending its identity and password to the application server over the protected channel. For example, protection can be provided with TLS/SSL. If the LDAP server is configured so that principals need to authenticate to it, then the application server sends the Bind request to the LDAP server using credentials of the client. If this authentication succeeds (the LDAP server sends Bind response message with success status), then the application server considers the client to be authenticated. The other possibility is for the application server to simply retrieve the client's identity and password from the directory and compare with those received from the client [18]. In this case the application server should authenticate to the LDAP server prior to the information retrieval, and, besides, the application server should be authorized by the LDAP server to perform those actions.

Enterprise Single sign-on

The Single sign-on (SSO) is a mechanism that enables a user to authenticate once to a system and reuse this authentication during the current session. Technologies that can be used to provide single sign-on can be divided into three classes [8]: ticket–based, cookie-based, and PKI-based.

In Ticket-based SSO a user first authenticates to the authentication service and receives cryptographic ticket, and then this ticket is used to authenticate to service providers. Kerberos is a typical ticket-based SSO system.

In Web-based environments Enterprise SSO can be achieved via cryptographically protected HTTP cookies. Sun OpenSSO Enterprise 8.0 uses this approach to provide SSO solution. The SSO process used by OpenSSO consists of the following steps [22]:

1. A user sends HTTP request to a service provider. This request is intercepted by the policy engine that protects the resource. After examining request and finding no HTTP cookie, the policy engine redirects the user to another URL for authentication.
2. The browser follows the redirect URL and issues the new HTTP request to OpenSSO Enterprise authentication service. The authentication service by using one of its authentication modules, for example LDAP authentication as described in the previous paragraph, validates user's credentials. The HTTP response containing cookie that carries encrypted session token is sent to the client. The HTTP response also contains redirect to the original location.
3. The browser follows the redirect and sends HTTP request to the service provider one more time. However, this time the request contains cookie with the session token. The policy engine intercepts request and checks the session token. The check is made by contacting OpenSSO Enterprise service. OpenSSO Enterprise decrypts the token and checks whether the session data associated with the session token exists. The policy engine receives a response stating whether the token is valid. After the session token validation, the policy agent decides if the user should be granted access.
4. When the next time the user contacts some other service provider, the cookie with the session token is included in the request. So, the policy engine that intercepts the request needs only to validate the token. The second authentication procedure is not required from a user.

Public key –based SSO requires the usage of public key infrastructure. The trusted third party certification authority is responsible for checking user's credentials and issuing certificates. The certificate and the user's private key can be stored on a user's station, on a smart card, or on some cryptographic token [19]. The authentication of the user is made by the service provider itself, the certification authority checks identity of the user only when issuing certificate. The PKI infrastructure also provides nonrepudation service, which is quite important for business [8].

Identity federation

The business-to–business communication nowadays is characterized by extensive use of internet technologies. Business processes require having external connections with partners, suppliers, contractors, clients, etc. However, creating and managing accounts for external users locally is not an efficient solution from management, security, and operational cost point of view [20]. Identity federation is a concept that enables inter-organization identity sharing and management, and secure external access to a defined set of company's resources.

Identity federation relies on the trust relationship between collaborating organizations. It means that one organization trusts in authentication that was made by another organization for

some user. The trust relationship makes Single Sign-On service for cooperating organizations possible. An Identity provider is an organization that is responsible for maintaining and managing end users identity information. A service provider is an entity that provides some services. A circle of trust consists of at least one identity provider and a group of service providers that trust this identity provider. Many organizations have a role of service provider and identity provider at the same time [21], but for some companies it can be beneficial to outsource identity management to an identity provider.

Federation enables seamless interaction between organizations with completely different, independent environments. Collaborating organizations neither need to have similar security systems nor to have detailed knowledge of systems used by a partner [10]. There are several frameworks that define federated identity management: Liberty Alliance Identity Federation Framework (ID-FF), Security Assertion Markup Language (SAML) framework, Web Services Federation (WS-Federation). SAML v2.0 should be considered as a preferred solution [22] since Liberty ID-FF and SAML v1.x were contributed to OASIS consortium and formed the foundation of SAML v.2.0, and WS-Federation is an alternative solution for interaction with Active Directory Federation Services [22].

SAML [23] is an XML-based framework for exchanging identity, corresponding attributes, and authentication information between collaborating organizations. This exchanged information is expressed in the form of SAML assertions. Assertion contains a set of statements about a principal (user, computer, company, etc.). The SAML assertion can contain the following types of statements:

- authentication statements: created by the party that successfully authenticated a user; describe authentication mechanism, time of authentication, etc.
- attribute statements: contain specific principal's attributes (e-mail, tel.number)
- authorization decision statements: describe authorized actions

SAML assertions are transmitted in SAML protocol messages (both assertions and protocol messages are XML documents). And SAML bindings describe how protocol messages are carried by underlying transport protocols. SAML defines HTTP-based and SOAP-based bindings. SAML protocol does not provide security protection for message exchange [24]. It relies on other protocols like TLS/SSL or IPSec to provide confidentiality and integrity protection. Security can also be ensured by usage of XML encryption and digital signatures.

The main use cases for SAML are: Single Sign-on and Federated identity. A typical SSO scenario is when a client by using a browser application sends request to a service provider. The service provider redirects the user to the identity provider for authentication. After authentication, the identity provider issues assertion, which is used by the service provider to check whether the user should be granted access to resource. Any subsequent requests to other service providers in the same circle of trust do not require the user to go through authentication procedure one more time.

Identity federation model depends on many factors. For example, some factors that influence identity federation are: whether users have existing local identities, whether identity attributes about users should be exchanged, whether identity federation should be based on temporal identifiers that are destroyed after session termination, etc.

Some of the use cases described by SAML for identity federation [23]:

- Federation via Out-of-Band Account Linking: identity federation is established without usage of SAML protocol. For example, it could be done via database synchronization.
- Federation via Persistent Pseudonym Identifiers: permanent SAML pseudonym identifier is used to dynamically establish identity federation during web SSO exchange

- Federation via Transient Pseudonym Identifiers: A temporary identifier is used to temporary federate identity till the user's web session termination. The benefit of this approach is that an organization does not need to manage local accounts for users from a collaborating organization.
- Federation Termination: removal of an existing federation.

Identity federation enables organizations to provide access to external users from collaborating organizations without the need to manage these accounts locally, thus reducing administrative costs, simplifying identity management, and enhancing security. Besides, it provides benefits for external users presenting web single sign-on service. And SAML is standard solution that offers both identity federation and single sign-on.

Identity and access management is crucial for business environments. It is a mechanism that protects enterprise resources from unauthorized access initiated by inside as well as external users. Business objectives dictate the intense usage of internet-based applications and demand limited, secure access to enterprise resources for external users from collaborating organizations like partners, contractors, clients, etc. The variety end complexity of systems used by business complicates the identity and access management tasks. Centralized storages for identity and account-related information as well as centralized authentication make things more manageable. However, it is not always possible to have a single directory because of administrative, security or some other issues. That is why many identity and access management suites are being developed to provide centralized interface for management purposes.

2.1.3 Identity and access management in UMTS systems

The Universal Mobile Telecommunications System (UMTS) is a 3rd generation 3GPP mobile network technology that evolved from the GSM. It introduced enhanced security, a completely new radio access network that allowed much greater speeds then in GSM, IP as a transport protocol between network elements over the whole system (3GPP Release 5), etc. [27].

The UMTS network has two parts: the Radio Access Network (RAN) and the Core Network (CN). The CN in turn consists of the following domains [27]: the Packet Switched (PS) domain, the Circuit Switched (CS) domain, the Broadcast (BC) domain, and the IP Multimedia Subsystem (IMS).

The User Equipment (UE) consists of the mobile phone and the Universal Integrated Circuit Card (UICC) that contains Universal Subscriber Identity Module (USIM) application. The UICC may also contain another application called IP multimedia Services Identity Module (ISIM) that is required for services in the IMS. The UE has radio connections with Base stations (BS) that are the part of RAN. The primary task of BSs is to deal with radio signal receiving/transmitting [27]. Base stations (also called Node Bs) are connected to the Radio Network Controller (RNC) that implements most of the controlling logic of RAN.

Main components of the PS domain of the core network are the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). The SGSN is responsible for packet routing and transfer, it also maintains data needed to perform these functions [27]: temporary identities, location information, IP addresses, etc. The GGSN takes care of interconnecting with external IP networks.

The CS domain consists of the Mobile service Switching Centre (MSC) with the Visitor Location Register (VLR), which is a temporary database of the subscribers that are currently in the location area controlled by the MSC, and the Gateway MSC (GMSC). The main function of the MSC is a call control and switching of calls. The GMSC is responsible for interconnection

with telephone networks.

The Home Subscriber Server (HSS) is a main storage for subscriber data and identifiers. Among other functions, HSS is responsible for [27]:

- storing of addressing information for mobility management;
- security information generation and access authorization;
- call/session establishment support.

Equipment Identity Register (EIR) is another database that contains information about user equipment and its status (e.g. list of stolen phones).

The following diagram depicts UMTS architecture.

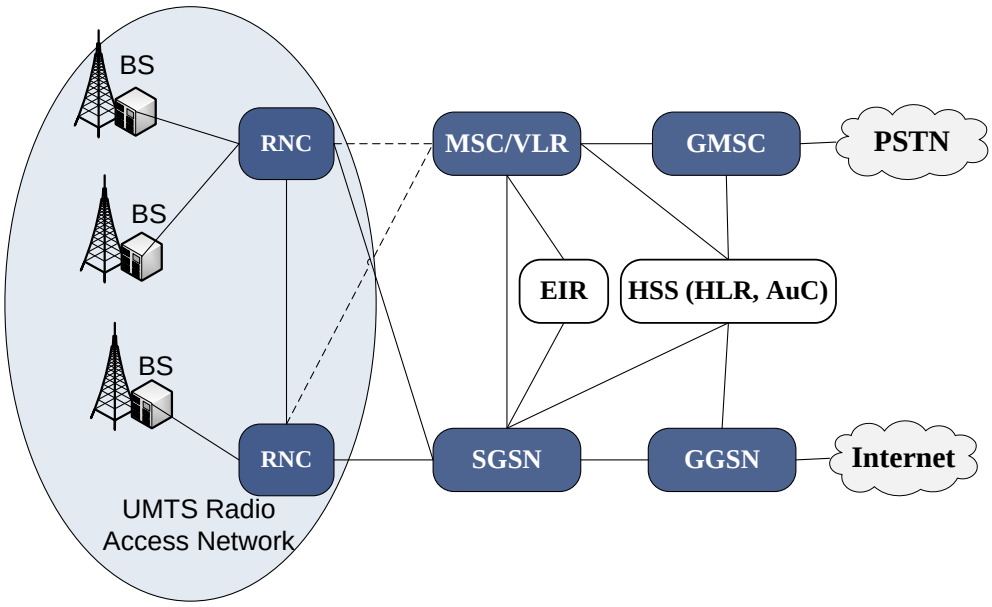

Figure 2.1.3.1: UMTS architecture

UMTS identifiers

UMTS has many different identifiers. Some are to ensure uniqueness of the subscriber identity, other are for security purposes.

The International Mobile Subscriber Identity (IMSI) [28] is a globally unique identifier for a subscriber. Consists of the Mobile Country Code (MCC), the Mobile Network Code (MNC), and the Mobile Subscriber Identity Number (MSIN) that is unique within the mobile operator's network. It plays a role of a search key in HLR, VLR, AuC and SGSN [27]. IMSI is also stored within USIM application in the UICC.

The Temporary Mobile Subscriber Identity (TMSI) and P-TMSI – are used to protect user confidentiality from passive eavesdroppers in CS domain and PS domain respectively. Subscriber identification in UMTS networks is in most cases performed by using these temporary identifiers instead of IMSI [29].

The Mobile Subscriber ISDN Number (MSISDN) is a globally unique identifier that acts as a telephone number. It is used for routing calls to the phone [30]. The MSISDN is stored in the HSS.

The International Mobile Equipment Identity (IMEI) is a globally unique number that identifies the mobile phone. The UE provides this number to the network (when requested) and the network verifies the status of the mobile phone by checking whether it is not blacklisted in

the EIR [27].

The IP Multimedia Private Identity (IMPI) is used by the IMS to identify subscribers. This identifier is stored within the ISIM application and in the HSS [29].

The IMS Public Identity (IMPU) – an identifier used by the IMS for addressing purposes [27]. IMPI is not intended for this [29]. There should be at least one IMPU associated with IMPI.

UMTS security

UMTS provides the following security services [29]:

- Mutual authentication between the serving network and the UE
- Confidentiality protection of user plane data (no integrity protection)
- Integrity and replay protection of signaling data
- Privacy protection

The UMTS mutual authentication is a secret shared key authentication. The 128-bit master key is stored both in USIM and in AuC.

The general authentication scenario happens when the user enters the area serviced by some other operator (serving network). The following steps take place [29]:

1. The subscriber identifier IMSI, or TMSI, or P-IMSI (P-TMSI) is transferred to the VLR or SGSN (in case of packet switched connection);
2. The serving network sends authentication data request to the AuC in the home network;
3. The AuC generates authentication vectors (secret key is one of the inputs for generation) and sends them to the serving network. Authentication vector consists of {random number (RAND), expected response (XRES), encryption key (CK), integrity key(IK), and the authentication token (AUTN)};
4. The serving network (MSC/VLR or SGSN) sends authentication request to the UE. The request contains the AUTN to authenticate the network to the UE and the challenge RAND;
5. These parameters are transferred to the USIM. The USIM authenticates the network and generates RES – response to the challenge and temporary keys CK and IK (RAND and secret key are used as input);
6. The UE sends the RES value to the serving network. It is compared to the XRES, and if values match, the authentication is passed.

UMTS encryption is based on the stream cipher that uses block cipher KASUMI to produce pseudorandom bit stream [29]. The secret key used in KASUMI is 128-bit long (CK), and the input and output blocks are 64 bits.

UMTS integrity protection also relies on the KASUMI cipher that is used in slightly modified form of CBC-MAC mode [29]. The 64-bit output is truncated to produce a 32-bit MAC value. Both the encryption and the integrity protection take place between the UE and the RNC.

Privacy protection relies on the TMSI and P-TMSI identifiers that are used instead of the permanent identifiers. After initial registration, which utilizes permanent identifiers, the system generates temporary identifiers, and after encryption is started these temporary identifiers are sent to the UE [29]. These identitifiers are then used for all subsequent communications.

UMTS is interoperable with GSM. Mobile phones with SIM cards can be used in UMTS (under the assumption that these mobile phone support UMTS RAN technology) and mobile phones with USIM can be used in GSM. When the UMTS capable mobile phone with UICC card that contains USIM applet is used in the GSM network we have 2G encryption and authentication [29]. When the UMTS capable mobile phone with UICC card that contains USIM applet is used in a 3G network with GSM base stations, we have 2G encryption but 3G authentication [29]. This means that interoperability leads to the decrease in the security strength. The protection is maximal only in the pure 3G scenario without any interoperation.

2.2 Smart card technology

Smart card technology is crucial for modern information systems. The deployment of smartcards is enormous because of the widespread usage of smart cards in mobile phone networks, international payment systems like MasterCard and Visa, transport and ticketing systems. Besides, smart cards are widely utilized in identification and access-control systems. The reason of this success lies in the security services provided by smart cards.

2.2.1 Smart card definition and types

A smart card can be defined as a card that has a size of a credit card and contains embedded integrated circuits. Thus we are not taking into account magnetic stripe cards that can be used only for information storing. Magnetic stripe cards do not provide sufficient level of security and can be easily forged [31].

There are several classes of smart cards with different functionality and purpose:

Memory chip card: This type of cards acts as storage for information. They generally have no on-board processing facilities. Memory chip card provides almost no security gains compared to the magnetic stripe card [31, 32]. Although the memory chip card can contain one-time-programmable memory that is written once and cannot be rewritten later [33], in contrast to the magnetic stripe card, it is still easy to read the stored value and produce the copy of the card [31, 33].

More sophisticated memory cards are wired logic-integrated smart cards that in addition to write/erase protection restrict read access [35]. They have few fixed extra functionalities and the available small command set can be changed only by redesigning the chip [32]. Although arithmetic logic unit is very limited, it is able perform simple cryptographic operations for authentication and data encryption. For example, MIFARE classic provides authentication of the reader based on the stored keys with encryption of all subsequent memory operations [34].

Microprocessor chip cards: This type of smart cards contains a microprocessor, an operating system, several types of memory, and I/O circuits. Optionally smart cards can also contain a crypto coprocessor to accelerate execution of cryptographic operations. Smart cards contain the following memory types: Read Only Memory (ROM), Random Access Memory (RAM), and Electrically Erasable Programmable Read Only Memory (EEPROM). ROM contains data that is stored there during the manufacture process and that cannot be modified, it can only be read during card operation. The operating system of a smart card is stored in ROM. RAM is a volatile type of memory. It is used as a dynamic data storage that loses its content on power shutdown. EEPROM is a non-volatile memory, thus data is saved when power is removed. EEPROM is used as a general storage for data and application program codes. One of the major drawbacks of EEPROM memory is its limited number of write cycles, although it can be read unlimited amount of times. Other memory types like flash memory with shorter write access time and longer lifetime compared to EEPROM are slowly gaining popularity in smart

cards [31].

The rest of the overview is primary devoted to the microprocessor chip cards.

Smart cards have two distinct chip card interfaces: contact and contactless. Contact and contactless smart cards are standardized in ISO/IEC 7816 and ISO/IEC 14443 standards respectively. The main difference between these two interfaces is that a contactless reader produces energizing radio frequency field for energy transfer to the contactless smart card via air, and the modulation of this field enables transfer of data. Therefore, contactless smart cards also have an embedded antenna. There are also dual-interface cards that have both contact and contactless interfaces.

The operational frequency for contactless operation defined by ISO/IEC 14443 is 13,56 MHz. Contactless smart cards are further divided into proximity and vicinity smart cards. Proximity smart cards have a limitation that they must be in a close proximity (up to 10 cm) from a reader to function properly. ISO/IEC 15693 standard describes vicinity cards. The operational range for vicinity cards is up to approximately 1 m [33]. Supported data rates according to ISO/IEC 14443 for contactless smart cards are: 106, 212, 424, and 848 Kbit/s [31]. Although vicinity cards provide greater operational distance, the data transfer rate is lower than for proximity cards and is only 26.48 kbps [31].

The wireless interface introduces an additional vector of possible attacks on smart cards which include but are not limited to attacks like eavesdropping, denial of service, man in the middle. However, according to [35], contact and contactless cards can basically provide the same level of security if threats specific to contactless interface are taken into account in the security architecture of a smart card.

2.2.2 Security provided by smart cards

One of the fundamental functions of a smart card is to provide secure storage for data [36]. Thus, a non-modifiable memory plays a significant role in a smart card security, since it can be used as storage for system secret keys [33] (typically top keys of the key hierarchy). Another approach is instead of storing the system secret key in ROM, is to compute it based on an unique chip serial number that is stored in the ROM [31, 33].The secure microprocessor is a heart of the smart card security system. The word “secure” mainly means that microprocessor is protected against physical and side-channel attacks. The introduction of microprocessors in smart cards occurred primary because of security reasons, since microprocessors made cryptographically protected communications possible [35]. Without it a smart card would just have features of an ordinary memory chip card. From the security point of view the functions of microprocessor include: generation of pseudo-random numbers for crypto protocols, generation of temporal keys, encryption/decryption, digital signature generation, performing operating system security checks, for example evaluation of whether access to smart card resources should be granted, etc. Therefore, besides providing secure storage, smart cards can be used for secure execution of cryptographic algorithms [33].

Unlike memory chip cards, an access to microprocessor smart card’s resources is controlled by an operating system that is run on a microprocessor. Consequently, the logical level security is provided by the operating system access control system that grants or denies access to smart card resources. Some of the security mechanisms provided by modern smart card operating systems are listed below:

- Access control: the access to smart card resources is based on specific access conditions and is allowed only to authorized entities. Access control can be based on a state-oriented or a command-oriented access model [33]

- Authentication: card reader authentication, user authentication (prior to allowing any operations on behalf of the user, the user should be authenticated) , authentication of communicating parties
- Process isolation: a process can't access resources owned by other processes
- Atomic transactions: either all of the operations that constitute a transaction are performed or none of them
- Secured communication: smart cards utilize security protocols for communications to protect information during transfer.
- Cryptographic protection: smart cards support usage of various cryptographic mechanisms like encryption, hashing, digital signature, random number generation.
- Key management: the operating system is responsible for secure generation, distribution, usage and destruction of cryptographic keys
- Security monitoring and audit: the system events can be monitored and analyzed on the presence of potential security violations
- Secure data deletion: data after deletion should not be accessible or recoverable
- Card locking: ability to temporary/permanently disable a specific application or the whole card

However, an attacker can try to obtain data directly from the memory or buses during data transfer, avoiding the security checks made by the operating system. Therefore, to mitigate this threat smart cards also provide protection against physical attacks.

Physical attacks on smart cards can be divided into invasive and non-invasive attacks. The invasive physical attacks require removal of the microprocessor from a card to get direct access to it. This kind of attacks requires sophisticated equipment (ex.: a microscope, laser, micromanipulators, focused ion beams) and a lot of technical knowledge [31, 33]. Invasive attacks start with an analysis of the microprocessor structure. That is why the first line of protection measures is deployed at this stage to complicate the analysis for an attacker. The protection measures can be the following: a small size of IC components make it hard to extract information [33, 35]; the layout of the chip's blocks can be random [31], dummy structures on the chip that have no meaning to the functions of the chip [33], covering the chip with a metal layer hides the layout of the chip [31]. Although hiding the structure of the cheap will make analysis harder for an attacker, it should not be considered as an adequate protection against physical attacks. The following mechanisms provide stronger protection specifically against intrusive attacks:

- Buried and scrambled chip buses: Buses are buried inside of the chip to prevent direct contact. Besides, buses are scrambled to complicate understanding what these buses are responsible for [33, 35].
- Current-carrying metal layer on top of the chip: Shielding metal layer on top of the whole chip can be current-carrying. Besides protecting from internal structure analysis, this metal layer also protects against attacks that use electrical measurements from the chip's surface [33]. Consequently, if it is removed, the chip will not be operational.
- Memory encryption: Encryption of the volatile and non-volatile memory and some microprocessor registers, with some specific keys [33, 35]. Even if the data is read from memory, it will still need to be decrypted.
- Buses encryption: encryption of data in buses during transfer [35]
- Anomaly sensors: The sensors are used to detect abnormal environmental conditions. Voltage monitoring, external clock monitoring, temperature monitoring are examples of available sensoring techniques [31, 33, 35]. It is important to note that clocking is provided by external source, consequently the processing speed of a smart card is also regulated by this external clocking source. It could be extremely

beneficial for an attacker to lower the frequency drastically to make measurements easier [33]. Therefore the clocking sensor should be implemented to protect against this attack.

Side-channel attacks are an example of non-invasive attacks. In the side-channel attack a device is being monitored during its normal operation in order to obtain some leaking information. Timing information, power consumption levels, electromagnetic leaks that are correlated with execution of some commands/computations on secret data can reveal some information about this secret data. The timing attack measures execution time of particular operations to get some knowledge about secret data. The power analysis can provide information about which operation and with what parameters is being executed based on the power consumption levels. The Differential power analysis (DPA) is even more powerful technique since the statistical analysis is applied to power measurements done while repeatedly processing known data and then unknown data to notice difference in the power consumption for different input data [31, 33]. As a result the unknown data can be revealed either partially or fully. The electromagnetic analysis deals with electromagnetic radiation from a chip, but the principle is the same.

The usage of constant execution time algorithms which have the same execution time for different input values protects against timing attacks. However, microprocessors that have constant power consumption due to hardware tweaks are very expensive and are not practicable [38]. Random delays used in an algorithm and other more sophisticated masking techniques can decrease/destroy correlation between measured parameters and secret data [31, 38]. Besides, according to [38], most of these protection measures are weak against differential fault analysis.

Differential fault analysis is based on injection of faults to disturb the operation of a microprocessor. Abnormal voltage, clocking, temperate, and electromagnetic influence can lead to skipped commands, misinterpreted commands, data read with errors [31]. Countermeasures include [31]: checksums, execution of the same commands several times and comparison of results, variable redundancy when a variable has a copy that is modified along with the original and then they are compared, etc.

Smart cards are complex devices and security is achieved by combination of logical and physical protection measures. A security hole in one of these two layers can result in partial or complete compromise of a smart card. Thus careful design of software and hardware is crucial for smart card security.

To evaluate and assure a security level provided by smart cards, a smart card's claimed security functionality must be tested against some international standard. Common Criteria is an international security evaluation standard that checks the fulfillment of the security functionality of a product against some requirements. Evaluation is done by independent certified laboratories. The depth and the scope of the evaluation defines an evaluation assurance level (EAL) issued by a certifying organization. Thus EAL represents the level of confidence that the security functionality of a product meets the requirements [37]. There are seven assurance levels ranging from EAL1: "functionally tested" to EAL7: "formally verified design and tested". Protection profiles describe a set of security features that must be provided for a specific product type. A developer of a product determines at which EAL the evaluation is made against a set of security requirements.

A smart card has three distinct layers/modules: integrated circuit, operating system, applications. These components can be evaluated separately or as a whole system [35]. The modular approach (integrated circuit, OS, and applications are evaluated separately) is more favorable since the change in one module means reevaluation for that particular module and not for the whole platform [31, 35]. Besides, after the modular security evaluation a composition evaluation (IC, OS, and application as a one system) can be done [31].

Security is a cornerstone in a smart card development. The combination of logical and

physical security mechanisms that form a unified system, which optionally can be evaluated according to international standards, ensures a high level of security. This ability to store information and execute cryptographic protocols in a secure manner provided a great success for smart cards in security sensitive areas.

2.3 Smart card OS and 3GPP applications

The first smart card platforms were application specific. They were designed primary for one application, had a monolithic architecture, and were embedded in ROM [31, 33]. Thus it was quite problematic to introduce any modifications after the card was manufactured. The next generations of smart cards was characterized by decoupling of applications from the operating system. The introduction of multi-application smart cards to the mobile communication environment led to separate naming for a hardware part and a software part of the smart card. The hardware layer with low-level software is called the Universal Integrated Circuit Card (UICC), and the application for mobile telephony is called the Universal Subscriber Identity Module (USIM).

The UICC card, besides the USIM application, can hold other applications like an application for mobile banking, an application for mobile payments, etc. The SIM application for GSM networks, the USIM application for UMTS networks, and the IP Multimedia Services Identity Module (ISIM) application for the IP Mobile Subsystem can reside on one UICC.

Some of the modern commonly used open standard multi-application smart card platforms are: Multos, Java Card 3, and GlobalPlatform. Both Multos and Java Card can contain USIM (SIM, ISIM, etc.) application, although the technology is quite different. Multos is a complete multi-application operating system while Java Card is just a platform that must reside on top of some operating system.

The UICC card with the USIM application is owned and managed by a mobile operator. GlobalPlatform is a framework that enables secure post-issuance application management. Because of the enormous number of subscribers, the only possible and feasible solution for mobile operators to manage issued cards is Over-The-Air (OTA) management [31]. The OTA management is commonly performed via SMS bearer. However, because of its limited capacity another method that uses the Bearer Independent Protocol (BIP) with USIM application toolkit (USAT) is preferred.

2.3.1 Java Card 3

The Java Card platform provides a secure multi-application execution environment for Java applications on a smart card. The Java Card technology optimizes Java language and Java runtime environment for smart cards. It was specially designed for constraint-based chips and does not have all functionality of the standard Java technology. However, the continuous progress in the smart card manufacturing allows to gradually increase the platform functionality. For example, the Java Card Platform Connected Edition provides support for multithreading and concurrent execution of applications; it also supports Web applications that can interact with off-card clients via HTTP/HTTPS [42].

The Java Card platform consists of the Java Card platform virtual machine, the Java Card platform runtime environment, and the Java Card application programming interface (API). The central part of the Java Card platform is the Java Card virtual machine that is a runtime Java bytecode interpreter. It is responsible for the applet execution and security. The Java Card virtual machine acts as an abstraction layer between the underlying OS and applets executed by this virtual machine. Thus, the smart card operating system's complexity is hidden, which simplifies application development [31], and the portability of Java applications is provided. Figure 3.4.1 depicts the Java Card architecture.

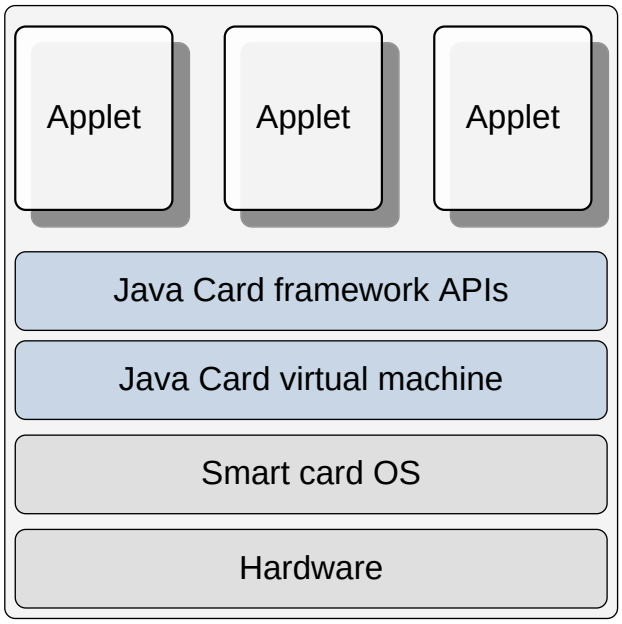

Figure 2.3.1.1: Java Card architecture

The Java Card virtual machine is never terminated. It works till the end of the lifetime of the smart card [39]. The Java Card virtual machine stops temporary when the power supply is removed, and it resumes its work and state when it is powered on again [39]. The Java Card virtual machine's lifetime starts when it is burned with the smart card operating system into the ROM of the smart card [31].

The Java Card platform security is ensured by the following security features [40]: the transaction atomicity, cryptographic classes, and the applet firewall. In addition, secure applet loading, installation, and deletion is provided.

Transaction atomicity means that a data modification occurs only if a transaction was completed normally and fully, otherwise the transaction is canceled and the card returns to its state before transaction started.

The applet firewall is a mechanism that provides applet isolation. Every applet receives a designated memory area and no other applet can access that area unless specifically allowed by the applet that owns that memory area [41]. It means that objects can be shared by applets and applets can interact, but it should be explicitly allowed.

The Java Card security and cryptography classes support [40]: public and private key cryptography, digital signatures, message digests, random number generation, and PIN management. This, for example, enables Java Card applications to communicate securely over the network with off-card peers using SSL/TLS.

The Java Card platform allows applet management after a card was issued. The following post-issuance functions are supported: loading, installation, and deletion of applications. The security is ensured by checking the digital signature of a new applet. Besides,

the Java Card platform allows delegating some card management functions to card management frameworks like GlobalPlatform.

2.3.2 GlobalPlatform

GlobalPlatform is a framework for smart cards that allows to simplify and accelerate development, deployment, and management of applications. It specifies an open and interoperable infrastructure for smart card systems. GlobalPlatform enables post-issuance smart card customization by reconfiguring and loading/installing new applications to the smart card under the full control of the card issuer. Thus, GlobalPlatform smart cards can be modified to perform various operations.

The GlobalPlatform architecture consists of the following elements [43]:

- Runtime environment: includes a smart card operating system, a virtual machine, and an Application Programming Interface (API). For example, the Java Card platform deployed on some smart card operating system can form a runtime environment for GlobalPlatform;
- GlobalPlatform API: consists primary of APIs that enable an off-card entity to communicate securely with a Card Manager and use services related to the card application management, for example locking the card [43];
- Card Manager: The Card Manager is the main controlling entity in the GlobalPlatform framework [31]. It allows the issuer of the smart card to have full control over the smart card. All communication with off-card entities is received by the Card Manager which is responsible for dispatching of received commands to appropriate applications. The Card manager is also in charge of controlling the content of the smart card. It keeps track of all installed applications. Loading and installation of new applications must be allowed by the Card Manager [43]. The Card Manager also provides cardholder verification mechanisms like the Global PIN code [31]. Besides, it is responsible for managing secure communication channels when some sensitive off-card communication is done. Each application is assigned a security domain. The Card Manager serves as issuer's security domain that controls issuer's applications on the card. Issuer's security domain holds issuer's secret keys, provides secure communication channels, and managers issuer's applications;
- Card applications: include various applications that can be managed by the issuer, or the management can be delegated to application providers [31];
- Security domains: security domains protect applications or group of applications. Security domains allow an application/service provider to control its applications on its own. Each service provider has its own security domain. Delegated management with security domains allows the application provider to perform loading, installation, and deletion of its applications. However, the issuer must authorize the application provider to perform these content management operations. Authorization is done by granting the application provider a cryptographic token that contains a digital signature of the issuer. When the application provider needs to manage its applications it presents the token to the Card Manager.

GlobalPlatform allows customization of a smart card in a secure manner so that the issuer always has full control over the smart card. It is becoming de facto the standard solution for loading and managing smart card applications [31]. Besides, it is able to work on top of any smart card operating system. These features have resulted that as of October 2009 approximately 450 million GlobalPlatform-based smart cards have been deployed worldwide [44].

2.3.3 Multos

Multos is an open standard multi-application operating systems for smart cards. It is an extremely secure architecture. Multos achieved ITSEC E6 level certification that corresponds to Common Criteria EAL7 [45].

The Multos architecture consists of [46]:

- Multos operating system: OS is responsible for input/output, file management, cryptographic services, etc;
- MULTOS Application Abstract (or Virtual) Machine: Multos Executable Language (MEL) is a byte code language that is interpreted by the virtual machine. Applications for the Multos architecture are developed using high level languages like C or Java and then compiled into the MEL byte code. The Multos virtual machine lies between the OS and applications. Thus it enables interoperability of applications between different Multos implementations [31];
- MEL API: MEL APIs for high level programming languages provide interface for application development independent from a hardware platform;
- applications: various MULTOS application programs.

Multos security system provides the following features [45]:

- only the card issuer has control over loading and installation of applications onto the card. The issuer can provide trusted third parties with cryptographic certificates that will allow them to load applications;
- an application cannot access the memory of another application. Execution-time checking of bytecode instructions ensures that illegal instructions or accesses attempts to memory areas of other applications are rejected by the virtual machine. Data sharing between applications is prohibited [47];
- existing applications are not effected by loading or removing of other applications;
- application loading mechanism ensures authenticy, integrity, and confidentiality of application data.

Convergence of mobile, security, and banking applications on one card requires a high level of security, and Multos is currently one of the most secure platforms. Unlike Java Card and GlobalPlatform, it does not need any underlying operating system since OS is defined in the Multos architecture. Besides, the Multos platform provides support for the post-issuance management of operator-specific applications via the OTA mechanism.

2.3.4 3GPP applications

Universal Subscriber Identity Module (USIM) application resides on the UICC card. It contains the logic that controls subscriber – network mutual authentication and cryptographic key agreement, secure storage of mobile service related information (secret keys, subscriber identity, etc.), and communication with a mobile equipment (ME). The UICC card contains an operating system on which the USIM application is running. Besides the USIM application, UICC can support other 3GPP applications like ISIM, SIM, USIM Application Toolkit, etc [48]. However, if the UICC contains both the USIM and the SIM application, then the ME shall always use the USIM application regardless of the radio access technology [50]. It means that a possibly existing SIM application will never be used if the USIM application is present.

During the network operation phase messages exchanged between the ME and the 3GPP application are in the form of command/response pairs [48]. And the ME plays the role of the master and application plays the role of the slave in this communication.

The UICC card file system has a hierarchical tree structure. It consists of the following elements [49]:

- Master File (MF) - the root of the file system;
- Dedicated File (DF) – enables functional grouping of files. It acts as directory. An Application DF (ADF) is a DF that contains all the DFs and EFs of an application;
- Elementary File (EF) - a file that contains various data.

Standards specify UICC file system access control mechanism and security environment, which provides protection for applications by assigning security containers for every activated application. The UICC security architecture consists of the following elements [49]:

- Security attributes: a set of access rules. Security attributes are attached to an ADF/DF/EF files;
- Access rules: consist of the set of requirements that must be met in order to perform operations on a file. Access rules define what operations can be performed and which security conditions must be met before performing these operations;
- Access Mode (AM): indicates to which operations the security condition applies;
- Security Condition (SC): specifies which security procedures (for example PIN verification) must be made before a command may be performed on a file.

Since the USIM application must be able to work with various mobile equipment platforms, it was comprehensively standardized and conformance tests were defined [31]. The USIM application specification among other things defines [50]:

- file structures;
- contents of EFs;
- security functions;
- application protocol to be used on the interface between UICC (USIM) and ME.

The USIM application stores all information in EFs, except for the shared secret key used for authentication. EFs can be mandatory, optional, or conditional. A conditional file is mandatory if it is required by some supported application feature. Some of the categories of information that the USIM application stores in EFs are: service-related information, phone book and call information, messaging information, location information.

Some of the important files defined by the USIM application specifications are provided below.

Table 2.3.4.1: USIM application files

File name	**Description**
EF_{IMSI} (IMSI)	contains the International Mobile Subscriber Identity (IMSI). The file is mandatory.
EF_{MSISDN} (MSISDN)	contains The Mobile Station International Subscriber Directory Number (MSISDN). This file is optional.
EF_{SPN}	contains the service provider name. The file is optional.
EF_{Keys}	contains the ciphering key CK, the integrity key IK and the key set identifier KSI. The file is mandatory.
EF_{KeysPS}	contains the ciphering key CKPS, the integrity key IKPS and the key set identifier KSIPS for the packet switched (PS) domain. The file is

	mandatory.
EF_{UST}	Indicates which services (SMS, MMS, Local phone book, GSM Access, etc.) are provided by the USIM. The file is mandatory.
EF_{LOCI}	contains the following Location Information: • Temporary Mobile Subscriber Identity (TMSI); • Location Area Information (LAI); • Location update status
EF_{PSLOCI}	contains the following Packet Switched location information: • Packet Temporary Mobile Subscriber Identity (P-TMSI); • Packet Temporary Mobile Subscriber Identity signature value (P-TMSI signature value); • Routing Area Information (RAI); • Routing Area update status.
EF_{SMS}	This file is present if SMS service in USIM is available This EF contains information concerning received/sent short messages
EF_{NETPAR}	contains information about network cell frequency parameters
EF_{NCP-IP}	This file is optional. It contains network connectivity parameters for USIM IP connections
EF_{Kc}	This file is present if "GSM access"service is available. It contains the GSM ciphering key Kc and the ciphering key sequence number n.
EF_{KcGPRS}	This file is present if "GSM access"service is available. It contains the GPRS ciphering key KcGPRS and the ciphering key sequence number n.

USIM standards specify different procedures that can be used by the mobile equipment to interact with the USIM. The ME can read/modify/delete information that is related to mobile communication service and stored on the UICC by interacting with the USIM application. Some of the procedures are described bellow [50]:

- USIM management procedures: include USIM intitializataion, 3G session termination, USIM application closure, USIM service table request (reading of EF_{UST}), etc.
- USIM security related procedures: include Authentication algorithms computation, IMSI request, Access control information request, reading/updating of cipher and integrity key, reading/updating location information, etc.
- Subscription related procedures: creation/deletion of information in the USIM phone book, reading/updating/erasing SMS messages from the USIM, etc.

USIM provides the following set of security features [50]:

- authentication of the USIM to the network;
- authentication of the network to the USIM;
- authentication of the user to the USIM;
- data confidentiality over the radio interface;
- file access conditions;
- conversion functions to derive GSM parameters.

A 3GPP application uses two Personal Identification Numbers (PINs) to authenticate a user [48]: level 1 verification requirements (PIN) and the level 2 verification requirements (PIN2). The PIN has a global access scope. Any file on the UICC that has the PIN referenced in access rules is accessible with this PIN [49]. It becomes an application PIN if it is assigned to that application. The PIN2 has a local scope. Besides these two PINs there is also a Universal PIN that is shared among several applications in multi-application cards. Application can use the Universal PIN as a replacement for the PIN.

Another application that can reside on the UICC is the IP multimedia Services Identity Module (ISIM) application. The IP Multimedia Subsystem (IMS) is a framework which combines data, voice, and video communication services on a single IP network. It is an open standardized architecture independent of access technologies (mobile, wireless, fixed network) that allows fast and easy introduction of new services [51]. This flexibility is accomplished by the introduction of an intermediate layer between the operator core network and applications. IMS architecture specifies three distinct layers: transport, session control, and application layer. It utilizes media gateways to convert different multimedia services into the IP domain. Session Initiation Protocol (SIP) is used for sessions (data, video, voice) management and registration of subscribers. Application servers provide services to customers.

To use IMS services a subscriber should be registered and authenticated. The ISIM is responsible for these operations in a networks with IMS architecture. It basically has the same functions as USIM application in UMTS architecture.

Some of the important files defined by the ISIM application specifications [52] are provided below.

Table 2.3.4.2: ISIM application files

File name	**Description**
EF_{IMPI}	contains the IMS private user identity of the user
EF_{IMPU}	contains one or more public SIP Identity (SIP URI) of the user
EF_{DOMAIN}	Contains the Home Network Domain Name

IMS identification is based on IMPI/IMPU identifiers and authentication is based on the shared secret stored in ISIM [51]. The ISIM supports Authentication and Key Agreement (AKA) protocol, defined in RFC 3310, for mutual subscriber - network authentication and generation of keys for IPsec encryption and integrity protection of the SIP messages between the mobile equipment and the SIP proxy server [53].

2.3.5 USIM application toolkit and SATSA

The USIM application toolkit (USAT) provides mechanisms which allow applications, existing on the UICC, to interact and operate with the ME [53]. It allows the USIM to initiate actions to be performed by the ME. The requests/commands that can be carried out are standardized. Some of the proactive actions are the following:

- displaying text or multimedia messages from the UICC to the terminal;
- sending short messages;
- setting up a voice call to a number held by the UICC;
- request ME's local information (IMEI, current time, language settings, etc);
- initiating a dialogue with the user;

- requesting the terminal to report geographical location information to the UICC

However, to use these mechanisms USIM must first determine capabilities of the ME. This is done during UICC initialization when the ME sends a profile with supported features. If no profile is sent, then UICC assumes that USAT is not supported.

One of the most powerful commands is CALL CONTROL [31]. Before setting up the call, all dialed digits and supplementary call control information is passed to the USIM [53]. The USIM can allow, modify, or block the call. It could, for example, be used to assist the user with dialing numbers by adding the country code automatically [31].

USAT supports Bearer Independent Protocol (BIP) that allows USIM to download data through the ME's high speed data bearers (Bluetooth, GPRS, 3G, etc) instead of using slow SMS bearer.

However, ME vendors do not provide sufficient support for the USAT and prefer to implement all features in the ME [31]. The handset developers are more interested in exploiting USIM functionality by mobile applications residing on the ME. For example, the Security and Trust Services API (SATSA) allows J2ME applications to communicate with smart card devices, including Java Card platform cards and UICC cards with USIM [54]. SATSA also provides security APIs for the management of digital signatures, digital certificates, and cryptographic operations.

J2ME application can use the functionality of the UICC card such as the phone book or applets. SATSA allows exploiting smart card security features like: secure data storage, cryptographic primitives, and secure execution environment.

2.4 Bluetooth technology and security

This section describes the Bluetooth technology, protocols, and studies authentication and encryption mechanisms provided by Bluetooth.

2.4.1 Bluetooth technology and protocols

Bluetooth is a low power short-range wireless communications technology. It operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 GHz. Bluetooth emulates full-duplex transmission using a time-division duplex (TDD) scheme [55, 56]. The Core Specification defines a minimum connection range of 10 meters. The specified connection range for class 1 Bluetooth radios is 100 m, and manufactures can exceed this value since there is no range limitation [57]. Bluetooth version 2.0 EDR supports data rates up to 3 Mbps, version 3.0 HS supports up to 24 Mbps.

Bluetooth connection utilizes master-slave relationship when one of devices acts as a master and the other as a slave [55, 56]. The master is responsible for providing synchronization parameters. A group of devices synchronized in this way forms a piconet.

The Bluetooth core system protocols are the Radio (RF) protocol, Link Control (LC) protocol, Link Manager protocol (LMP) and Logical Link Control and Adaptation protocol (L2CAP) [55, 56]. The Service Discovery Protocol (SDP) is another extremely important service layer protocol that is required by all Bluetooth applications [55, 56].

The Bluetooth generic data transport architecture consists of three layers: physical layer, logical layer, and L2CAP layer [55, 56]. Physical layer has two sub-layers: physical channel and physical link. On top of the physical layer resides a set of logical links and channels in the following sequence: logical transport, logical link, and L2CAP channel. Physical link is formed between each slave and master to provide transmission of data. Logical links use the physical

link as a transport. One or several logical links are multiplexed on the physical link. Logical links support unicast synchronous, asynchronous, isochronous traffic, and broadcast traffic [55, 56]. Creation, modification, and termination of logical links are handled by link managers on peering devices. Link managers communicate via LMP.

L2CAP layer provides logical channels for the transport of higher service protocols and application data [55, 56]. L2CAP performs protocol/channel multiplexing, segmentation and reassembly (SAR), per-channel flow control, and error control [55, 56]. L2CAP can also provide Quality of Service (QoS) for applications [55, 56]. Channels can be both connection-oriented and connectionless. Connection-oriented L2CAP channels are for unicast traffic, while connectionless are for broadcast data transport [55]. Unicast channels may be uni-directional or bi-directional. L2CAP signaling channel is responsible for creation and establishment of connection-oriented channels and negotiation of changes for both connection oriented and connectionless channels. Channels are identified by channel identifiers (CID). Each end-point of the channel has its own CID.

L2CAP can operate in the following modes [55]:

- Basic L2CAP Mode: it is a default mode. It is equivalent to L2CAP specification in Bluetooth v1.1. It is used if no other mode is used.
- Flow Control Mode: PDUs are numbered and acknowledged. Missing PDUs are detected, but there are no retransmissions
- Retransmission Mode: PDUs are numbered and acknowledged. Missing PDUs are retransmitted.
- Enhanced Retransmission Mode: added in Bluetooth Core Specification v3.0 + HS [56]. It provides some enhancements to Retransmission mode.
- Streaming Mode: added in Bluetooth Core Specification v3.0 + HS [56]. PDUs are numbered, but they are not acknowledged. Used for real-time isochronous traffic.

According to [56] the Flow Control mode and Retransmission modes should be used only in case when one of the peers does not support the Enhanced Retransmission or the Streaming mode.

L2CAP is frame-oriented. Applications that do not need transmission of data in frames can avoid L2CAP layer and work directly with baseband logical links [55, 56]. Depending on the mode of operation, L2CAP PDUs have different structure. The information payload size ranges from 0 to 65535 bytes, though in some connection modes the upper bound can be slightly smaller, but no less than 65527 bytes.

The Bluetooth interoperability between different devices is provided for some specific service(s) and use case(s) [55, 56]. Bluetooth profiles specify subset of messages and procedures from Bluetooth specifications and corresponding use cases. Devices that want to interoperate should conform to some common profile. Generic Access Profile (GAP) defines general procedures that can be used to establish connections, discover devices, etc. It defines generic modes of operation that are not service- or profile-specific and that can be used by profiles referring to this profile [55, 56]. Devices that conform to some other profile must be compatible with devices compliant to this profile [55, 56]. It means that GAP ensures basic compatibility between all Bluetooth devices.

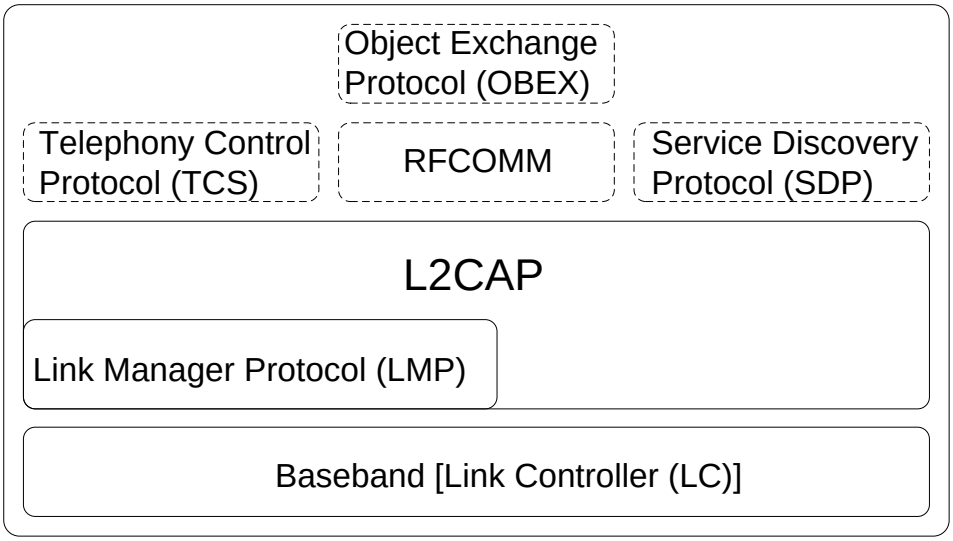

Figure 2.4.1.1: Profile stack covered by Generic Access Profile [55, 56]

SDP is a protocol for discovering services in Bluetooth environment and for determining characteristics of those services. It supports discovery of existing services without any prior knowledge of those services, besides, it also support searching of services based on the class of service or some specific attributes [55, 56]. SDP uses L2CAP as its transport protocol. SDP is a request/response client-server protocol. The client issues requests and the server answers them. The server has a list of service records, which describe services, associated with the server. One record describes one service. The service record is a list of service attributes. One attribute describes one service characteristic (e.g. ServiceName, ServiceClassIDList, etc.).

Each service belongs to some service class [55, 56]. The service class defines values for all attributes in the service record. Every service class has a 128-bit universally unique identifier (UUID). ServiceClassIDList contains a list of service classes of which the service is an instance. Attribute values in the service record must conform to all classes in the ServiceClassIDList simultaneously. Therefore, service classes contained in the list are related [55, 56].

Pre-allocated UUIDs have 16-bit UUID and 32-bit UUID aliases (short representation of the 128-bit value). It is impossible to search for services based on the values of arbitrary attributes; the search is based on UUIDs that describe the service [55, 56]. A service search pattern is a list that contains one or more UUIDs. It is used to find matching services. The service matches the search pattern if the record that represents the service contains all required UUIDs.

RFCOMM is a serial cable emulation protocol that provides transfer of serial data [58]. RFCOMM can establish up to 60 simultaneous connections between two Bluetooth devices. OBEX is a protocol that exchanges data objects that are defined by the OBEX protocol itself [59]. A Bluetooth device using OBEX is considered to be a client of the protocol. RFCOMM is a main transport protocol for OBEX.

2.4.2 **Bluetooth security**

Generic Access Profile defines a generic authentication procedure that specifies how the LMP-authentication and LMP-pairing procedures are used. It also describes 4 security modes (3 of them are legacy modes) that provide different levels of security [55, 56]. The legacy security

modes are for devices that support Bluetooth v2.0 + EDR and earlier:

- Security mode 1: is a non-secure mode that does not use any security mechanisms
- Security mode 2: is a service level enforced security mode in which services request security mechanisms. Device does not initiate any security procedures before the start of the L2CAP channel establishment procedure. Bluetooth device can specify the following security requirements:
    - Authorization required
    - Authentication required
    - Encryption required

  In case no security requirements are specified, this mode is equivalent to the first mode [55, 56].
- Security mode 3: is a link level enforced security mode. It means that security procedures are initiated before LMP completes link setup [55, 56]. This mode supports shared secret key authentication and encryption [60].

Security modes 1 and 3 are excluded in Bluetooth Core Specifications v2.1 + EDR and v3.0 +HS. Security mode 2 is used for backwards compatibility with legacy devices [55, 56]. Security mode 4 is the only non-legacy security mode defined in these specifications. It is a service level enforced security mode. Services can specify the following security requirements [55, 56]:

- Authenticated link key required
- Unauthenticated link key required
- No security required

The first two options utilize Secure Simple Pairing (SSP) procedure to create the connection. The *Authenticated link key required* uses either the numeric comparison, out-of-band, or passkey entry SSP association model. The *Unauthenticated link key required* uses just works SSP association model. By default, GAP chooses the unauthenticated link key and enabling encryption if both pairing devices support SSP.

Secure Simple Pairing

Secure Simple Pairing was designed to provide more user-friendly pairing procedure and improved security [55, 56]. It provides protection against passive eavesdropping and against man-in-the-middle attacks (in some association models) [55, 56]. Protection against passive eavesdropping is provided by Elliptic Curve Diffie Hellman (ECDH) public key cryptography used by SSP.

SSP procedure consists of five phases:

- Phase 1: Public key exchange
- Phase 2: Authentication Stage 1
- Phase 3: Authentication Stage 2
- Phase 4: Link key calculation
- Phase 5: LMP Authentication and Encryption

During phase 1 pairing devices generate ECDH public-private key pair, exchange public keys, and compute common Diffie Hellman key. The authentication stage 1 differs among SSP association models, all the rest phases are similar [55, 56].

Authentication Stage 1 (Numeric Comparison Protocol): It is used for scenarios when both pairing devices have a display and both are capable of having the user enter "yes" or "no" [55, 56]. In this stage each device generates a 128-bit nonce, and then devices publicly exchange them. Nonces are needed to prevent replay attacks [55, 56]. Then the commitment value, which cryptographically binds these nonces to the public keys exchanged in phase 1, is computed on each side. If commitment check succeeds, each device computes a 6-digit confirmation value and shows it to the user. Both devices should have displays to use this mode. The user should check whether the numbers are the same on both devices. If the values are the same it means that nonces were successfully exchanged between intended devices. This association model provides limited protection against man-in-the-middle attacks [55, 56].

Authentication Stage 1 (Just Works): used in scenarios when one of the pairing devices neither has a display nor a way to enter a 6-digit number [55, 56]. This association model utilizes Numeric Comparison Protocol, but never displays any numbers to confirm. The user may be just asked to confirm the connection. The *Just Works* protects only against passive eavesdropping. It is susceptible to the man-in-the-middle attack.

Authentication Stage 1 (Passkey Entry): the user inputs the same 6-digit passkey on both pairing devices. The other option is to generate the passkey on one of the devices and display the value to the user. The user is then required to enter this value into another device. This mode provides protection against the man-in-the-middle attack.

Authentication Stage 1 (Out of Band Protocol): this association model is used when some out-of-band mechanism is used to both discover the devices and to exchange cryptographic values used in the pairing process. In-band discovery of the peer device followed by the usage of out-of-band channel for transmission of authentication parameters is not supported [55, 56].

Authentication stage 2 is used to ensure that peers successfully completed exchange in Authentication stage 1 and to confirm the peering between devices. Unlike in Authentication stage 1, there is no user intervention in stage 2. The pairing devices generate two values based on the exchanged nonces in authentication stage 1 and the common Diffie Hellman key generated as a result of the first phase. Devices mutually exchange values and check whether they match.

In phase 4 devices compute common link key based on the Diffie Hellman key and the publicly exchanged data. In phase 5 pairing devices generate encryption key from the link key. This phase is the same as in legacy pairing.

In legacy security modes 2 and 3 the authentication is performed based on the PIN code. PIN is used as an input to compute initialization key that in turn is used to compute the link key. The PIN can either be a fixed number or a value selected by the user. The user can select the PIN and then enter it in each of the devices that are to be paired.

Authenticated link keys are stronger than Unauthenticated, and Unauthenticated link keys are considered to be stronger than legacy link keys [55, 56].

LMP authentication and encryption

Bluetooth LMP authentication is a simple challenge-response scheme. LMP authentication starts after the link key was derived. LMP authentication uses the link key as a shared secret to perform authentication. The verifier sends the claimant a 128-bit random value. The claimant takes received random value, its own Bluetooth address, and the shared link key and uses them as input to the block cipher SAFER-SK128 to produce authentication code [55, 56]. This code is sent to the verifier. For mutual authentication verifier and claimant switch rolls.

Bluetooth utilizes a stream cipher algorithm E0 for encryption. Each packet payload is encrypted separately. The keystream generation algorithm is based on the linear feedback shift registers (LFSRs). Three modes are available for baseband encryption [55, 56]:

1. No encryption: no messages are encrypted;
2. Point-to-point only encryption: broadcast messages are not encrypted;
3. Point-to-point and broadcast encryption: all messages are encrypted.

Bluetooth provides a basic data integrity check using 16-bit CRC code [55, 56]. The data integrity check is optional. The CRC field is appended to the end of the packet. CRC is computed over every octet of the packet header and packet payload. The payload is ciphered after the CRC is appended, thus the CRC field is encrypted [55, 56]. Although Bluetooth has a basic data integrity check, it is much weaker than cryptographically protected data integrity check with message authentication codes (MACs).

Another Bluetooth security problem is that the stream cipher algorithm E0 has some security flaws [60] and is considered weak [98].

Summary

Bluetooth Security mode 4 with SSP is the most secure solution for device pairing. Besides, this mode, unlike legacy modes, does not require the knowledge of the shared secret PIN from both sides. The “Just works” association model being susceptible to the man-in-the-middle attack cannot be used in environments where high security is required. Thus, we consider the SSP “Numeric comparison” and “Passkey entry” association models to be a suitable solution to establish a secure Bluetooth channel.

Bluetooth provides authentication and protection of user data by encryption of the packet payload. However, it does not provide cryptographic integrity protection and the encryption algorithm has some security flaws.

3. Analysis

In this chapter, the Windows logon architecture is analyzed. This chapter also contains an evaluation of existing authentication schemes based on mobile phones, and an evaluation of the Windows logon extensibility.

3.1 Authentication and authorization in Windows

This subchapter contains a study of security mechanisms in Windows, a study of authentication and logon process, and an analysis of Windows smart card architecture.

3.1.1 Security principals and access to objects

Security in Windows is based on the subject-object-actions relationship. When a subject requests to perform some actions on an object, the operating system checks whether the access should be granted based on security permissions associated with that object. According to [61], a subject (security principal) can be: a user, a computer, and a service (starting with Windows Vista and Windows Server 2008). An object can be [62]: a file, a device, a job, a process, a thread, shared memory sections, volumes, etc.

To make a decision whether to grant some principal access to an object, the operating system must know the identity of the principal. Therefore, prior to be able to access resources a user must authenticate to the system during the Windows logon process. Windows uses unique security identifiers (SIDs) to identify security principals. A SID is assigned during the logon process.

The structure of the SID:

$$S - 1 - 5 - \underbrace{21 - 2443930396 - 124871960 - 1960245352}_{\text{Domain identifier}} - \underbrace{1000}_{\text{RID}}$$

- S – SID designator. Consists of character "S";
- 1 – revision number of the SID specification;
- 5 – authority identifier. For SECURITY_NT_AUTHORITY the value is 5. Other possible values are: SECURITY_WORLD_SID_AUTHORITY, SECURITY_LOCAL_SID_AUTHORITY, etc;
- Domain identifier – identifies a domain or computer that issued SID;
- RID – Relative identifier. RIDs are used to ensure uniqueness of SIDs. RIDs for non-default users and groups start with 1000 and increase by 1 for every new principal.

All accounts (users and groups) in the Windows system can be divided in two classes: user-defined and automatically created by the system. Automatically created groups can be further divided in built-in and system groups. System groups have automatic and dynamic membership that is dependent on the principal's activity type. Built-in groups, on the other hand, are not much different from user-defined groups and are used to support a default Windows security model [63].

Some well-known built-in accounts in the Windows system.

Table 3.1.1.1: Built-in accounts in Windows

SID	**user/group**	**description**
S-1-5-*domain*-500	Administrator	Built-in system administrator account

S-1-5-*domain*-501	Guest	Built-in account that can be used to give limited access to those who do not have personal account
S-1-5-*domain*-502	KRBTGT	Service account used by Kerberos KDC
S-1-5-*domain*-512	Domain Admins	Members of this group are authorized to administer a domain.
S-1-5-*domain*-513	Domain Users	Includes all user accounts in a domain
S-1-5-*domain*-515	Domain Computers	This group includes client stations and servers in a domain
S-1-5-32-544	Administrators	Initially contains Administrator account as the only member. After the system joins a domain, Domain Admins group is added to this group.

Table 3.1.1.2: Well-known system groups in Windows

S-1-1-0	Everyone	This group includes completely all users
S-1-2-0	Local	This group includes users who log on to computer locally via connected terminal
S-1-5-2	Network	Contains users who log on via a network
S-1-5-4	Interactive	Includes users that log on interactively. Terminal server users are in this group, but they are not included in Local group [61].
S-1-5-5-X-Y	Logon Session	This SID is used to identify a logon session, not a principal. Each logon session of a user is assigned a unique ID (X and Y values are changed)
S-1-5-11	Authenticated users	Contains authenticated users

After a user is successfully authenticated, Windows creates an access token for that user. Access token contains the SID of the user, SIDs of all groups in which the user is a member (including built-in and system groups), the SID identifying logon session, privileges assigned to the user and to user's groups, etc. If privileges assigned to the user or to one of the groups in which the user is a member change, or the user membership in some group changes, then the user must relogin in order for the changes to take affect [63]. When the user relogins, a new access token that contains updated information is created. Every process and thread executed on behalf of the user gets a copy of the user's access token. When a process or a thread request access to some securable object they present an access token to the system for the access control check.

Every securable object in the system has a security descriptor associated with it. The security descriptor describes who can have access to an object and what kind of access is allowed. It contains the following information: the SID of the owner of the object, the SID of the primary group of the owner (according to [61], primary groups in Windows access control model are used only for POSIX compliance), a system access control list (SACL), and a discretionary access control list (DACL).

DACL defines users and groups and whether they are allowed to access an object. If an object does not have associated DACL then everyone can access it [65]. DACL consists of a

number of access control entries (ACEs). The ACE contains the SID of the subject, the flag that shows if access is allowed/denied, and an access mask that defines access rights for that subject. If DACL for an object contains no ACEs (empty DACL) then no one can access the object [67]. Windows has the following access rights [66]:

- generic access rights: include READ, WRITE, EXECUTE, and GENERIC_ALL that is combination of all three
- standard access rights: include DELETE – the right to delete an object, READ_CONTROL – the right to read object's security descriptor, WRITE_DAC – the right to modify DACL, WRITE_OWNER – the right to change the owner of the object, and SYNCHRONIZE right.
- SACL access right: the rights to get or set SACL in the object's security descriptor
- object-specific access rights: for a example, the right to create files in a directory is FILE_ADD_FILE

SACL controls whether attempts to access an object are logged. ACEs in SACL specify which access attempt types and by which subjects should be logged. Both failures and successful attempts are logged. Most objects in the Windows system do not have SACL [61].

In the discretionary access control model an owner of an object defines access permissions for other users and groups. In Windows system the owner of an object and the system administrator have a full control over the object. When the object has empty DACL no one (even the owner) can access the object, but the owner can still modify access permissions [61]. However, if the DACL of the object contains ACE with the SID S-1-3-4 "OWNER_RIGHTS", then implicit READ_CONTROL and WRITE_DAC rights of the owner are ignored. This mechanism was introduced in order to prevent users from modifying permissions for their own files [61].

When a process or a thread requests excess to a securable object, the system checks the provided access token against the DACL of the object considering the requested access type. ACEs in the DACL are checked sequentially for SIDs that match those in access token until access (everything that was requested) is granted or denied or the end of DACL is reached. It means that access is granted when one or more access allowed ACEs permit all requested access rights for any subset of SIDs in the security token of the principal [68]. Access is denied when the deny ACE, for one of the SIDs in the access token, that denies any of the requested access rights is encountered in the DACL [68]. If the end of the DACL was reached and there is some requested access right that was not allowed by ACEs, then access is denied.

Users of the Windows system have rights and privileges. A privilege is a right to perform system-related management operations like changing time, rebooting the system, loading drivers, etc. [69]. Thus, privileges differ from access rights that subjects have to securable objects. Privileges are held in the assigned access token whereas access rights are defined in the security descriptor of securable objects. It is important to distinguish user rights (logon rights) and access rights to securable objects. Logon rights are like privileges except the fact that they are used to allow a user to log on to the system, and privileges determine what a user can do after logon [61, 63]. Some of the logon rights are presented in the following table.

Table 3.1.1.3: Logon rights in Windows

Logon right	**Description**
Access this computer from a network	determines which users/groups can logon via a network. By default this right is granted to everyone.

Allow log on locally	defines who can interactively logon to the system via connected terminal
Allow log on through Terminal Services	defines who can logon to a remote computer via Remote Desktop Protocol [63]
Deny access to this computer from network	defines for whom network logon is denied
Deny log on locally	defines for whom local logon is denied

Role-based access control

The Role-based Access Control (RBAC) is an alternative to the discretionary access control system that is used in Windows. Access to resources is based on a role. It means that privileges are assigned to roles. The role is an abstraction that reflects which operations and job functions a user can perform. When a user is assigned to a role he obtains privileges of that role.

Windows Authorization Manager (AzMan) provides support for RBAC in Windows. Administrative tools allow to define role-based authorization policies against which access control check will be made.

3.1.2 Authentication and logon process

There are two authentication/logon models in Windows – interactive and noninteractive. The interactive authentication means that the user is prompted for credentials. The noninteractive authentication uses credentials previously entered by a user during interactive authentication. It means that the interactive authentication must always precede the noninteactive authentication. Noninteractive authentication occurs when a user requests connection to other stations/servers in the domain.

The interactive logon to the system can be either a local or a domain logon. In case of the local interactive logon a user gets access only to the local system resources, and in case of the interactive domain logon a user gets access to resources of the whole domain.

Logon process in Windows XP

Windows XP interactive logon architecture includes the following components:

- Winlogon process
- Graphical Identification and Authentication (GINA)
- Local Security Authority (LSA)
- Authentication packages (NTLM and Kerberos)

The Winlogon process is responsible for managing logon procedure. It ensures that no other illegal processes can intercept logon information supplied by a user [62]. Winlogon relies on the GINA dynamic link library to obtain user's logon information. After credentials are obtained GINA calls the LSA to authenticate the user by using one of the authentication packages. Authentication packages are DLLs that encapsulate the authentication logic. The result of authentication is returned to GINA which in turn returns it to Winlogon. Winlogon starts user's shell if authentication was successful.

The logon process starts with a user pressing CTRL+ALT+DEL - the secure attention sequence (SAS). Winlogon registers this sequence during the boot process and no other process can intercept this sequence instead of Winlogon [62]. When Winlogon detects SAS it calls GINA to obtain user's credentials. GINA provides interface to get credentials from a user.

Default GINA DLL (MSGina.dll) can be changed with a custom GINA in order to modify the standard interactive logon procedure. A custom GINA can, for example, communicate with an external device to get user's credentials. If default authentication packages are not able to analyze credential information obtained by the custom GINA, then a custom authentication package should be used. LSA supports custom authentication packages. The ability to use a custom GINA and authentication packages allows to implement virtually any authentication scheme.

Upon obtaining user's credentials, GINA calls the LsaLogonUser function to authenticate a user by using one of the authentication packages. LSA uses specified authentication package to authenticate the user. If the logon is local then the local LSA authenticates the user, otherwise, in case of the domain logon, the LSA on the domain controller does it [64]. Microsoft provides two packages for authentication: MSV1_0 authentication package for the local logon and the Kerberos authentication package for the domain logon. The MSV1_0 compares user name and hashed password with those stored in the Security Account Manager (SAM) database [62]. The MSV1_0 is also used in case of the cached domain logon. In this case cached credentials are stored in the LSA database in the encrypted form. MSV1_0 returns the result of authentication to LSA (that in turn returns it to GINA), and if authentication was successful creates a logon session. The Kerberos authentication package operates in basically the same manner except that authentication exchange is done via the network and the authentication decision is made on the domain controller.

In case of successful authentication, LSA checks local policy database for the logon rights of the user. If the user does not have appropriate logon rights for the logon method that was used, the logon session is terminated and the failure notification is sent to Winlogon [62]. If the user has appropriate logon rights then LSA creates access token with appropriate account SID, group SIDs, session SID, and a set of privileges retrieved from the LSA policy database. LSA returns authentication result to GINA, and GINA returns it to Winlogon. In case of successful authentication, Winlogon additionally receives the user's access token. After that Winlogon launches user's shell (default is Explorer.exe) and provides it with user's access token [63] so that the shell can perform operations on behalf of the user. From the shell other processes are created. These processes inherit the user's access token.

Logon process in Windows Server 2008, Vista, and Windows 7

The logon architecture in Windows Server 2008, Vista, and Windows 7 was redesigned so that GINA is not used. Instead, there is a new Credential Provider model [72]. Moreover, in earlier Windows versions a console session (session 0) besides being interactive logon session also hosted system processes and services [70]. In the new architecture session 0 became non-interactive, thus users log on to separate sessions starting from session 1. It means that services in session 0 are isolated from user applications. Thus, services that run with higher privileges are protected against attacks from a malicious application code [71].

The new interactive logon architecture includes the following components [73]:

- Winlogon process
- Logon user interface (Logon UI) process
- Credential providers
- LSA
- Authentication packages.

The GINA's functionality has been devided between Winlogon, Logon UI, and credential providers [74]. The Logon UI is responsible for loading the credential providers and displaying the Windows logon interface to users. Credential providers gather credentials.

Credential Providers are responsible for [73]:

- describing the credential information required for authentication
- handling communication and logic with external authentication authorities
- packaging credentials for interactive and network logon

They are not security enforcement mechanisms [73]. Security is enforced by LSA and authentication packages. Default credential providers support password and smart card authentication [72]. Credential providers are implemented as in-process COM objects located inside DLLs. They run in the local system context.

LSA authenticates a user by sending a request to an authentication package. There are six authentication packages [75]:

- Credential Security Support Provider
- Microsoft Negotiate
- Microsoft NTLM
- Microsoft Kerberos
- Microsoft Digest SSP
- Secure Channel

Authentication protocols are responsible for verifying credentials of a user, a computer, or a process. The authentication protocols are security support providers (SSPs) that are implemented in the form of DLLs [76]. Some protocols are combined into authentication packages.

An application that requests authentication can specify either some specific authentication package (used directly if both systems support it) or it can specify Negotiate package. Microsoft Negotiate is a security support provider (SSP) that negotiates the best SSP for authentication between parties [77] via the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) [78]. Currently the Negotiate SSP selects between two protocols: Kerberos and NTLM [wind auth doc]. If both parties support Kerberos, it is preferred over NTLM. The NegoExts (NegoExts.dll) extension to the Negotiate package negotiates the use of SSPs other than Kerberos and NTLM [79], for example the Public Key Cryptography Based User-to-User (PKU2U) SSP.

The Credential Security Support Provider protocol (CredSSP) is a SSP that delegates credentials for remote authentication. It uses TLS tunnel to transfer credentials.

The logon process starts when a user enters SAS. After SAS is entered, the Winlogon starts the Logon UI to provide the user interface for logon. The Logon UI starts all credential providers registered in HKLM\Software\Microsoft\Windows\CurrentVersion \Authentication\Credential Providers. The Logon UI calls credential providers to obtain credentials, then it passes these credentials to the Winlogon. It queries every credential provider for the number of credentials that provider wants to enumerate and for the number of UI fields required to display credentials. The Logon UI displays tiles to the user, after all providers have enumerated their tiles. The user chooses a tile and interacts with the credential provider via the tile to provide credentials.

After the credential information has been gathered on the tile, it is packaged into a buffer and passed to a calling application [80]. In case of the logon scenario, the Logon UI passes credentials to the Winlogon. The Winlogon calls the LsaLookupAuthenticationPackage function to get a unique identifier for some authentication package [72]. Then it calls the LsaLogonUser function and provides obtained credentials and the authentication package

identifier to authenticate the user via the specified authentication package. If authentication was successful, LSA checks in the local policy database whether the user has sufficient logon rights. Then it creates the access token and returns the authentication result and the access token to the Winlogon. If authentication was successful, the Winlogon launches user's shell.

3.1.3 Smart card logon architecture

Smart cards provide a two-factor authentication that is based on the possession of a smart card and the knowledge of a PIN code. A smart card securely stores a private key and a corresponding X.509 certificate with a public key. The private key never leaves the card. Besides, it is stored only on the smart card [81]. During authentication, the smart card performs cryptographic operations using the private key, thus proving to the authentication server that the principal's smart card holds the key. However, to perform cryptographic operations a user must first authenticate to the card. The user proves his identity to the card by presenting a PIN. It means that the user is prompted only for the PIN rather than user name, domain name, and password during the logon procedure. A workstation interacts with smart cards via smart card readers.

In Windows systems, smart cards can be used to log on only to domain accounts, not local accounts [52]. Thus, the standalone smart card logon is not natively supported in Windows systems, though there are some commercial products offering this functionality. With the domain smart card logon, even in the case of a network service disruption or a failure of the domain controller, it is still possible to logon to a workstation that belongs to that domain using an offline logon capability.

Smart card domain logon session for the Windows XP [81]:

1. A smart card is inserted into a card reader. The insertion of the smart card starts the logon process automatically;
2. The Winlogon calls GINA to obtain user's credentials. GINA presents a logon screen to the user. The user is prompted only for a PIN;
3. GINA sends received PIN to the LSA;
4. The PIN is used by LSA to access the smart card.
5. LSA calls Kerberos Authentication Package (Kerberos SSP). Kerberos SSP creates a Kerberos Authentication Service Request to the KDC that contains principal's certificate and a cryptographic signature generated with the corresponding private key for the Kerberos pre-authentication [82].
6. The KDC validates the certificate (verifies certification path, checks revocation status, etc.) and checks the digital signature. After making these checks KDC retrieves user account information from Active Directory. This information is used to construct a TGT. Authorization data fields in the TGT contain principal's SID, the SIDs for domain groups to which the user belongs, and (in a multi-domain environment) the SIDs for any universal groups in which the user is a member. The public key from the certifacte is used to encrypt symmetric encryption session key. The KDC’s digitally signed response among other things contains the TGT, the KDC’s certificate, and the encrypted session key. If the client possesses the private key that corresponds to the public key in the certificate then he will be able to decrypt the session key and use it for the subsequent interactions with KDC.
7. Upon receival of the response the client validates the KDC certificate and checks the digital signature. Using the private key the client can decrypt the session key for communication with KDC. In order to log on to the computer the Ticket

Granting Service (TGS) to the local computer must be obtained from KDC. The remaining part of the authentication procedure is the same as for a standard logon session.

After a successful domain logon, Windows caches credentials. Thus, it is possible for a user to perform local/offline logon to the computer with the domain account even if the domain controller or the network connection failed. However, during local smart card logon with cached credentials a Certificate Revocation List (CRL) check is not done [83].

Windows 7, Vista, and Server 2008 Smart Card Infrastructure

Unlike previous version of Windows, Windows Vista supports [74]:

- smart cards that contain several (not one as previously) certificates solely for the logon purpose, besides certificates for other purposes. The number of certificates that can be stored depends on the smart card memory space;
- changing the PIN and unblocking a smart card without the need to log on first with a standard user name and password.

In order for a smart card to work with Windows, it must have its own Cryptographic service provider (CSP) [74]. CSP is a module that performs cryptographic algorithms for authentication, encoding, and encryption.

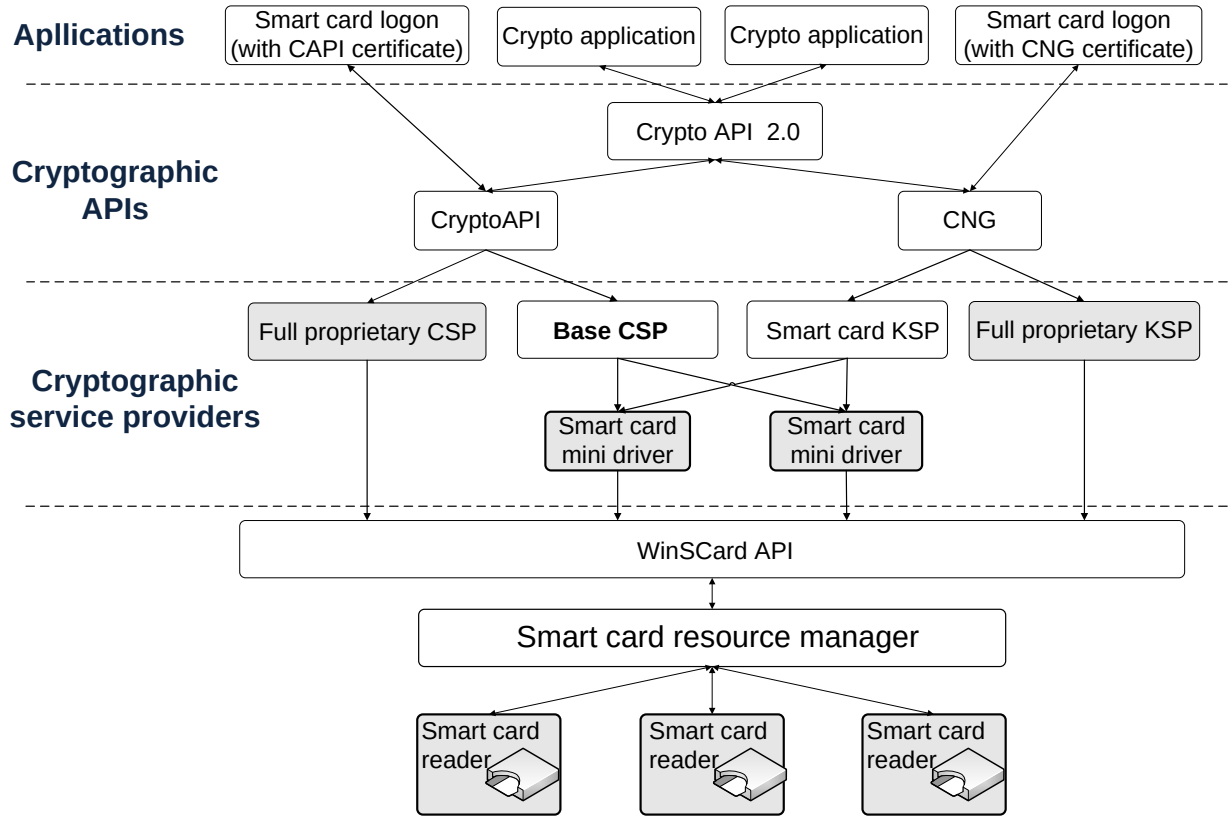

Figure 3.1.3.1: Windows 7, Server 2008 (R2), and Vista smart card architecture [74, 89]

Figure 3.1.3.1 depicts the smart card architecture in Windows 7, Windows Server 2008 (R2), and Windows Vista. The CryptoAPI contains functions that allow applications perform authentication, encoding, and encryption. To provide these services CryptoAPI functions use CSPs. The second generation of the CryptoAPI is called the Cryptography API: Next Generation (CNG). CNG, unlike CryptoAPI, separates cryptographic providers from the key

storage providers (KSPs) [90]. KSPs are responsible for managing keys. This includes creating, deleting, exporting, importing, and storing of keys. KSPs can also perform asymmetric encryption, secret agreement, and signing [90]. All cryptographic algorithms that are supported in CryptoAPI 1.0 are also supported in CNG [91]. Besides, CNG supports elliptic curve cryptography. CryptoAPI 2.0 provides a superset of CryptoAPI 1.0 that manages X.509 digital certificates [92].

The Base Cryptographic Provider is a CSP provided by Microsoft that makes it easier to write a smart card CSP [74]. It supports data encryption, hashing, and digital signature algorithms. Instead of writing a full propriety CSP, developers can write card-specific modules called smart card mini-drivers for the Base CSP. Although the Base CSP with custom smart card mini-drivers eases CSP development, it has a major drawback. Some of the algorithms provided by the Base CSP have limitations on the supported key size, and the offered key length values are inadequate from the security point of view. For example, the maximum key length for the RSA public key exchange algorithm is 512 bit, for the RC2 and the RC4 ciphers it is 40 bits (salt length is 88 bits) [93]. Applications built using CryptoAPI or CNG cannot change cryptographic algorithms implemented in providers [94].

The CNG Smart Card Key Storage Provider is another module that, like the Base CSP, is provided by default in Windows. The Smart card KSP, like the Base CSP, utilizes smart card mini-drivers. However, in terms of key lengths that are used in cryptographic algorithms [90], smart card KSP is not so limited like the Base CSP. RSA smart card mini-drivers can be registered with both the Base CSP and the smart card KSP [96]. However, only the smart card KSP can work with ECC-only as well as ECC/RSA dual-mode smart cards. To sum up, the CNG smart card KSP is a better choice in terms of security and should be preferred over the CAPI Base CSP for smart card based logon schemes.

The smart card KSP and the Base CSP implement commonly needed functionality. If developers require some additional functionality, then they should implement their own proprietary CSP or KSP.

A smart card mini-driver "*translates the characteristics of particular smart cards into a uniform interface that is the same for all smart cards*" [96]. The smart card mini-driver interacts with a corresponding smart card through the smart card resource manager that provides a common interface for communication with various smart card readers. The mini-driver emulates the logical card file system and provides a set of primitive capabilities [96] for the Base CSP/ KSP.

The association of the card-specific mini-driver with a smart card is based on the Answer-To-Reset (ATR) value that is received from the card [96]. The ATR value should be unique for every card. After the system receives ATR from the card, it looks in the registry for the ATRs that are stored there to find a match. The registry, actually, besides ATRs keeps also ATRMasks. These card-specific values are stored under the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Calais\SmartCards registry key. During the matching process the system iteratively goes through this registry key and applies the ATRMask to the ATR received from the card and compares the result with the ATR stored in the registry.

For the Personal Identity Verification (PIV) compliant smart cards [95] and for the Identity Device with Microsoft Generic Profile (IDMP) compliant smart cards [96] (beginning with Windows 7) Microsoft provides an inbox generic class mini-driver [96]. Presence of the PIV or IDMP compliant application on the card is used to associate a card with an inbox mini-driver. If no compatible inbox mini-driver is available, the Windows Plug and Play tries to install a smart card mini-driver that is logo-certified through the Windows Logo Program (WLP). For the Plug and Play scenario, a unique ID for the smart card must be derived. This unique ID is obtained either from the device ID or from the ATR. Thus, ATR is not always used

to associate a mini-driver with a smart card.

When a smart card is inserted into the reader, the following takes place [96]:

- Smart Card Plug and Play Process: If no compatible inbox mini-driver is available, the PNP downloads a logo-certified mini-driver from the Windows Update.
- Winscard Discovery Process: associates a smart card that is installed in the system with a PIV- or IDMP-compatible class mini-driver.
- Windows Smart Card Class Mini-driver Discovery Process: a card mini-driver marks the associated card as PIV- or Microsoft IDMP-compatible when the smart card is inserted into the reader and the Base CSP/KSP calls CardAcquireContext.

The following table shows AID values that are used in these processes.

Table 3.1.3.1: AID values [96]

AID name	AID value
PIV AID	A0 00 00 03 08 00 00 10 00 xx yy
Microsoft (MS) IDMP AID	A0 00 00 03 97 42 54 46 59 xx yy
MS Plug and Play AID	A0 00 00 03 97 43 49 44 5F 01 00

The Smart Card Plug and Play process in order to install a smart card mini-driver must first derive a unique ID for the smart card. The procedure consists of the following steps executed by the PNP:

1. PNP gets a smart card ATR value that may be used later in ID derivation
2. It sends a SELECT command to locate the MS Plug and Play AID.
3. It sends a GET DATA command to get a unique identifier.
4. If it fails to obtain a unique identifier, it tries to SELECT the PIV AID. The PIV-compatible device ID or ATR historical bytes are used as a unique ID.
5. If it fails to SELECT the PIV AID, it tries to SELECT the MS IDMP AID. The IDMP-compatible device ID or ATR historical bytes are used as a unique ID.
6. If PNP fails to select the PIV AID or the MS IDMP AID, it uses ATR historical bytes for the unique ID.

The Windows smart card class mini-driver performs the following steps to mark the card as PIV- or Microsoft IDMP-compatible:

1. It tries to locate the PIV AID by issuing the SELECT command. If the result is successful, the card is marked as PIV-compatible.
2. If it fails to select the PIV AID, it tries to SELECT the IDMP AID. If it succeeds, the card is marked as IDMP-compatible.
3. If it fails to select the IDMP AID, the card is still marked as the IDMP-compatible card and the system tries to work with it as if it were the IDMP-compatible card.

Smart card logon procedure in Windows 7, Vista, Server 2008 R2

The logon procedure consists of the following steps [74, 97]:

1. A smart card insertion does not start the logon. It starts only after SAS is pressed. The WinLogon requests the Logon UI to obtain credential information.
2. The smart card credential provider obtains a list of smart card readers and a list of inserted smart cards. For each card, it checks whether the logon certificate is present on the card. Found logon certificates are retrieved from the smart card and copied into a temporary secure cache. The smart card credential provider provides the Logon UI with logon certificates.
3. The Logon UI displays the logon user interface with found certificate logon tiles to a user. The user selects one of the tiles, and a PIN input box is displayed. The entered PIN is encrypted by the smart card credential provider.
4. The smart card credential provider returns the encrypted PIN, user name, etc. to the Logon UI. The Logon UI calls the LsaLogonUser function and provides the received data to the LSA.
5. LSA calls the Kerberos Authentication Package (Kerberos SSP) to create a Kerberos Authentication Service Request. The remaining part of the Kerberos authentication procedure is the same as in Windows XP.
6. If authentication is successful, then certificates are read from the card (including the root certificates) and stored in the user's certificate store (MYSTORE).
7. Upon removal of the card, the certificates are removed from the temporary secure cache. However, the certificates will still be present in the user's certificate store (MYSTORE).

According to [84], Windows 7 and Windows Server 2008 R2 have some minor enhancements to the smart card platform compared to Windows Vista that are primary related to the Plug and Play service and smart card drivers.

3.2 Evaluation of Windows extendibility: custom authentication mechanisms

There are three groups of components in the Windows logon architecture that can be customized:

- End-devices: smart card readers, biometric scanners, USB tokens, etc.
- Credential providers: password credential provider, smart card credential provider, and custom credential providers
- Authentication packages

The Winlogon, the Logon UI, and the LSA cannot be modified.

Optionally architecture can also include Windows services, which manage interaction between end-devices and applications/processes, and supporting APIs. For example, both the smart card infrastructure and biometric infrastructure have dedicated Windows services: Smart card resource manager and Windows Biometric Service respectively. They are responsible for tracking and managing all drivers that belong to their specific device group. All communication with end-devices goes through these services. These Windows services have supporting APIs that provide applications with access to their functionality.

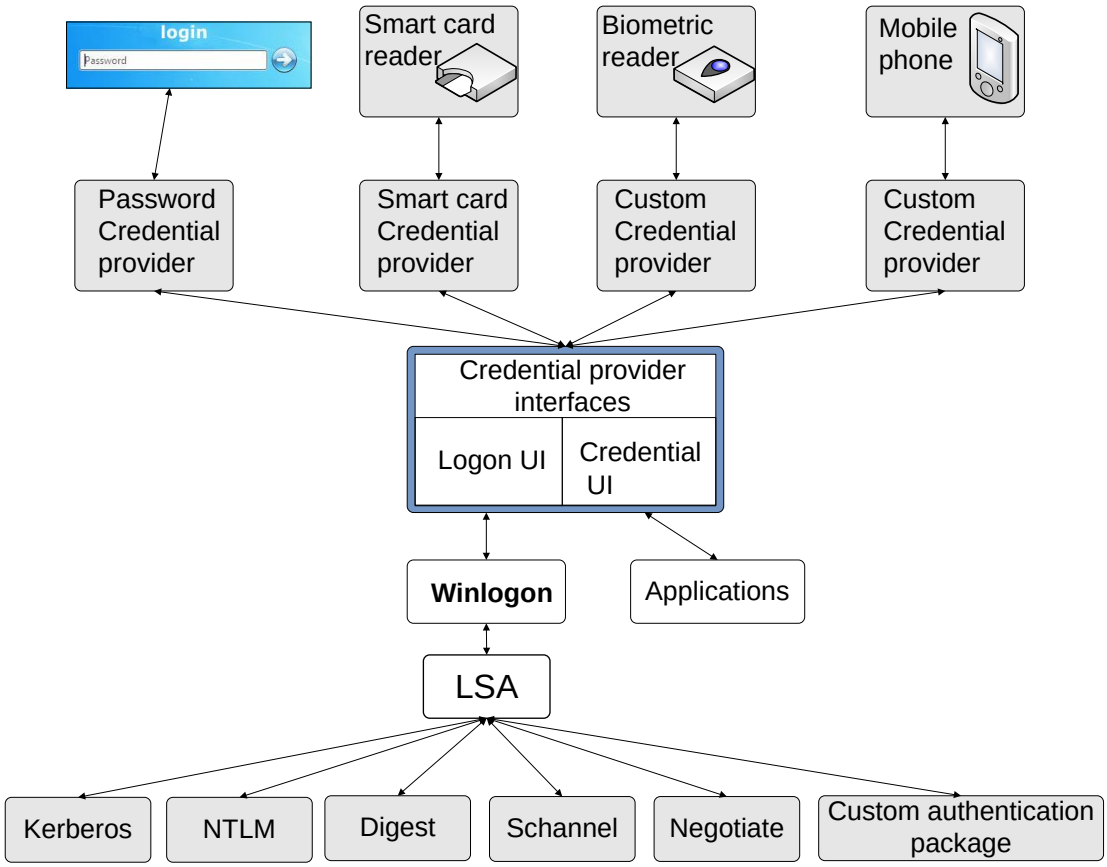

Figure 3.2.1: Windows authentication architecture

According to [85], logon customization is much easier and more secure in the new architecture with credential providers than in the old model where custom GINA DLLs had to be developed. GINAs required constant updating since each Service pack resulted in its break [80]. Previously GINA was responsible for the graphical rendering of the logon interface. In the new architecture, this is the responsibility of the Logon UI. The credential provider informs the Logon UI which graphical control elements (checkboxes, edit boxes, etc) are required to obtain user's credentials. If the Logon UI crashes for some reason, then the Winlogon will simply restart it [72]. The risk that the Winlogon will crash because of the bugs in a credential provider code is mitigated. Another problem of the old architecture is that a code written for authentication at logon did not naturally extend to authentication in Credential UI [80].

It is possible to install multiple custom credential providers. Custom credential providers can be developed to support different authentication mechanisms. Either a user can decide which credential provider to use, or the selection of the credential provider can be event-driven. It is possible to make obligatory for all domain users to use a custom logon mechanism [73].

Credential Providers can be used for multiple scenarios. They can be designed to support Single Sign On (SSO), application-specific credential gathering, authenticating users to a secure network access point, and the workstation logon [80].

Credential providers are quite flexible in terms of supported GUI controls for interaction with a user. They can specify (do not render) the usage of the following GUI elements: checkboxes, combo boxes, editable text fields, buttons, etc.

If you only need to extend the credential information that the existing credential provider collects, then it is possible to write a credential provider that reuses the functionality of the existing credential provider instead of re-implementing it [86]. This process is called “wrapping”. Most of the calls are proxied through the existing wrapped credential provider, and

the results are returned as if the wrapping credential provider produced them. The calls to the introduced extensions should be processed by the wrapping credential provider itself.

If default authentication packages are not able to analyze credential information obtained by a custom credential provider, then a custom authentication package should be used. The LSA supports custom authentication packages. The custom security package API supports development of custom security support providers for noninteractive authentication, as well as custom authentication packages for interactive authentication [87]. The LSA Logon Functions are used to access interactive authentication services. For noninteractive authentication services, the Security Support Provider Interface (SSPI) can be used to directly access services, thus bypassing LSA. Every authentication package must implement a specific set of functions that LSA calls when it receives authentication requests [88].

The ability to use custom credential providers and authentication packages allows implementing virtually any authentication scheme.

If we consider some general logon scheme as an extension to the current logon infrastructure, then depending on the authentication protocol used we may have one of the following scenarios:

- Neither existing credential providers no existing authentication packages fit our needs;
- The existing credential provider lacks some required functionality (e.g. does not collect some information), however an authentication package that implements the required authentication protocol already exists;
- The existing credential provider collects all required credentials and implements all required functionality, but there is no authentication package that implements the authentication protocol that we need;
- There is already a credential provider and an authentication package that are completely sufficient for us.

The first scenario requires the most work: both the credential provider and the authentication package should be implemented. The second scenario means writing either a wrapping credential provider or a completely independent credential provider. The third scenario requires implementing of a custom authentication package. The last scenario utilizes existing components. However, if the authentication solution utilizes some custom device, then a driver for the device should be written. This is also true for all other scenarios. Even if drivers already exist for the devices used in the authentication solution, it may happen that they do not provide all the functionality needed. Then it would also mean writing device drivers.

Considering mobile phone based authentication schemes, it is impossible to design a credential provider for some generic case that involves all existing and possible mobile phone based schemes. Credential providers are task specific. They are designed to work with some specific credentials and collect them in some specific way. Although it would be possible to write a Windows service that would provide an interface to the credential provider to deal with any device that belongs to the "mobile phone device group" in a predefined way, it still would not cover all possible variants since we do not know what credentials might be used in the future. And even if our mobile phone based generic credential provider would be able to receive these credentials, without knowing anything about how the data was obtained from the mobile phone, it would not be able to process unknown credentials. The usage of Windows services is justified only in frameworks (e.g. biometric framework) that cover many different devices from different produces that interact with framework in a predetermined way to provide interoperability. And even in the biometric framework there is no "one fits all" credential provider, developers need to develop their own. Only in the smart card architecture we have a

credential provider that deals with any smart card that conforms to PC/SC specifications, but that is because smart cards function strictly according to these specifications.

For example, many mobile phones nowadays have a micro-USB connector. Therefore, they can be used as USB tokens for some authentication scheme. One could write a credential provider and a corresponding Windows service that would track such devices and provide interface for the credential provider to access these devices, but it is easier to write an event-driven credential provider that would react to the insertion of such phone in the computer and interact directly with the phone.

USB and Bluetooth are two communication interfaces that are very common nowadays. Almost all mobile phones support Bluetooth, many in addition support micro-USB. Therefore, in this paper these connections are considered as primary for mobile phone based logon/authentication schemes. It must be noted that USB connection has some obvious advantages compared to Bluetooth. First of all, USB connection offers higher security since it is almost impossible to eavesdrop data transferred over a short USB cable between the mobile phone and the computer. The same is true for man-in-the-middle attacks. Besides, there is no need for authentication mechanisms as in Bluetooth, you see where you plug-in the cable. Secondly, USB has higher transfer speed. Lastly, USB does not have common radio media problems like interference.

The mobile phone with micro-USB connection can be seen as a USB token. It would be possible to write a credential provider that would automatically detect USB insertion, check whether it is an appropriate device, and read credential information. Then this credential provider would pack credentials for one of the existing authentication packages or for some custom authentication package. This case corresponds either to the first or to the second extendibility scenario.

If the mobile phone based authentication scheme relies on smart cards (utilizes the smart card inside the phone) and uses existing authentication protocols, then the usage of already available smart card infrastructure with the smart card credential provider seems to be a quite straightforward choice. This case corresponds to the 4th extendibility scenario when neither a custom credential provider nor a custom authentication package need to be implemented. Mobile phone in this case acts as a USB or Bluetooth smart card reader depending on the used connection. One of the architectures proposed in this thesis follows this scenario. The other proposed architecture corresponds to the third scenario in which the custom authentication package should be implemented.

3.3 Evaluation of existing mobile phone based authentication schemes

In this section, the existing mobile phone based authentication schemes are described and evaluated. In most of these schemes, a mobile network operator (MNO) has a role of an identity management (IDM) provider. All of these schemes in one or another way rely on the authentication of the phone (UICC) to the MNO. Some of these schemes use only Internet channel for authentication, but most of them utilize two channels: the Internet channel and the GSM/UMTS channel.

In authentication schemes that use two devices (in case of logon procedure it is a computer and a mobile phone) and multiple channels for authentication protocol exchange (the Internet connection from the computer to the authentication server and the GSM/UMTS connection from the mobile equipment (ME) to the authentication server) it is crucial to ensure that the same user controls both devices [99].

3.3.1 **SMS authentication with session-ID check**

A mobile network operator (MNO) has a role of an IDM provider. This authentication scheme is based on the fact that a user is already authenticated in the GSM/UMTS network. Therefore, the authentication process consists of the steps that ascertain that the owner of the ME is the same user that controls the computer. The verification is done by sending a sessionID both to the computer over the Internet and to the ME over the GSM/UMTS network. The ME owner or the ME itself (automatically) checks that the sessionIDs are the same and sends a confirmation message via SMS to the authentication server. The authentication procedure is completed when the AS receives the confirmation SMS.

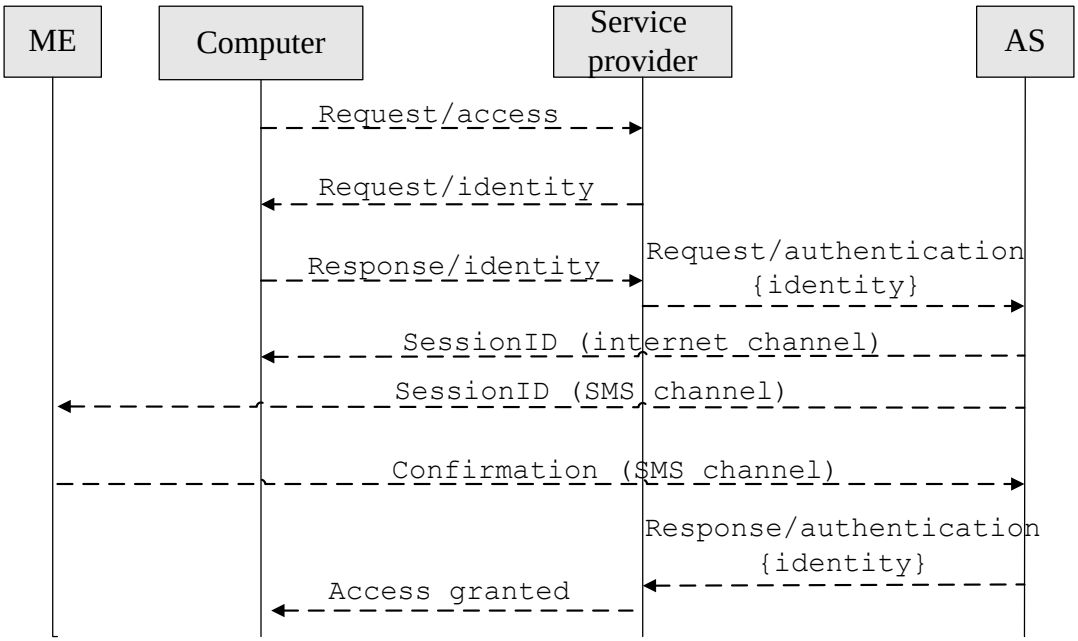

Figure 3.3.1.1: SMS authentication with sessionID check

A service provider is a client of the authentication server (AS), it outsources authentication to the AS. For MNOs, the HSS (AuC) is the authentication server.

For automatic verification of sessionIDs by the ME, a Bluetooth connection between the computer and the ME is required.

The described authentication scheme does not provide mutual authentication, it only authenticates the user to the service provider. The security of this scheme relies on the security of the underlying GSM/UMTS network, and is stronger in case of the UMTS network. This scheme does not have any explicit integrity protection. SMS forgery threat is mitigated by the GSM/UMTS security mechanisms. However, it is still possible to spoof SMS sender's address [99].

The following attack can be easily executed if the attacker controls one of the intermediate nodes between the user's computer and the service provider. When the user starts authentication to the server provider the attacker also starts authentication (supplying the identity of the victim) to the same or the other service provider that uses services of the same AS. The attacker blocks the original request from the user so that it does not reach intended service provider and responds with a forged Request/identity to the user. When the attacker receives sessionID via Internet channel from the AS, he "forwards" it to the victim. The user will receive two similar sessionIDs: one forwarded by the attacker via Internet channel and the other one sent by AS via SMS. The user (the ME in case of automatic verification) has no way to check that the received sessionIDs were actually issued by the AS to provide authentication service for the computer with different internet address and possibly for the different service

provider. Therefore, the user would issue confirmation SMS, and the AS would reply to the service provider that authentication was successful.

In case of the automatic sessionID verification by the ME using the Bluetooth connection to the computer, it is possible for the attacker to authenticate to any service provider registered at the AS even without the user noticing it (unless the PIN is required to issue confirmation message), if the Bluetooth security is compromised. Though the attacker would need to be in the close proximity to launch this attack.

3.3.2 SIM Strong authentication

The SIM Strong utilizes the EAP-SIM protocol to provide mutual authentication [100]. The MNO has a role of the IDM provider. Thus, the service provider relies on the MNO to perform authentication. The EAP-SIM authentication provides higher level of security compared to GSM authentication. The EAP-SIM provides the following enhancements to the GSM Authentication and Key agreement (AKA) procedure [101]:

- 64-bit long GSM encryption key Kc is used for deriving Master key and is not directly used
- multiple authentication triplets are combined to create authentication responses and session keys of greater strength than the individual GSM triplets
- Transient EAP Keys for protecting EAP-SIM packets, a Master Session Key for link layer security, and Extended Master Session Key are derived from the Master key

EAP-SIM provides mutual authentication, confidentiality, integrity, and replay protection.

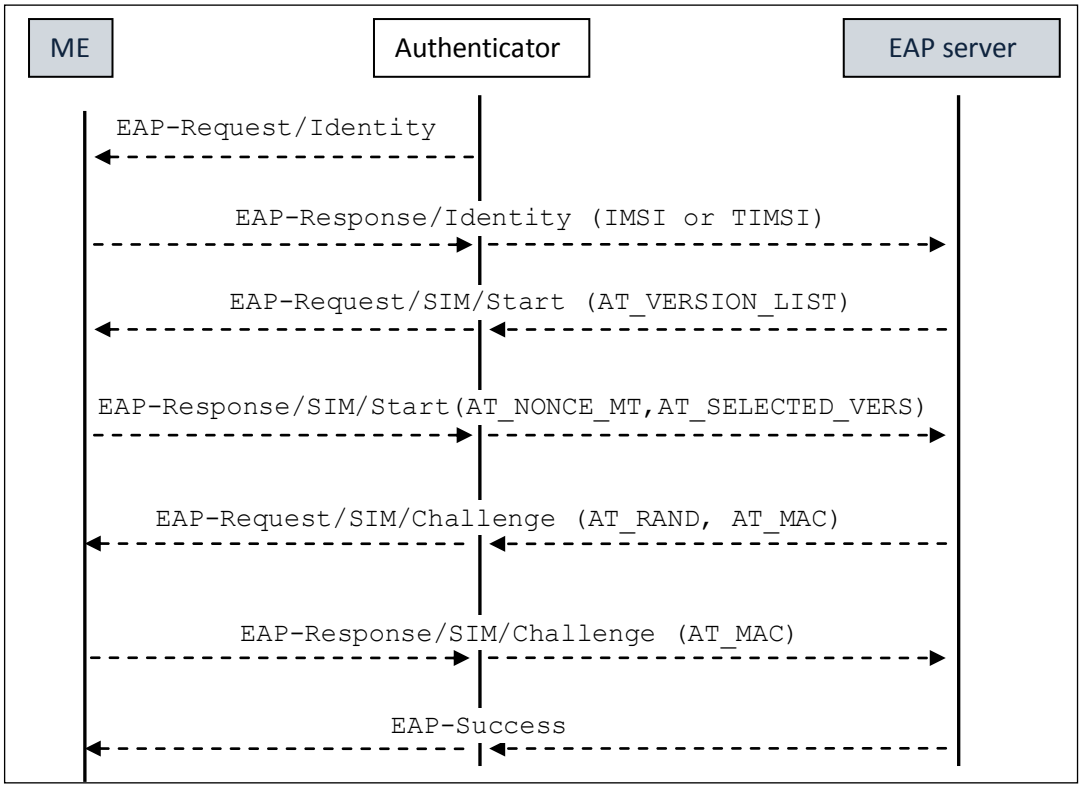

Figure 3.3.2.1: EAP-SIM authentication

The SIM Strong authentication can be performed either via the Internet + Bluetooth channel or via the SMS channel with the usage of sessionIDs [100]. In case of the Internet + Bluetooth SIM Strong authentication, the ME has the Bluetooth connection with the computer

and the EAP-SIM authentication exchange is performed via the Bluetooth channel and the Internet channel, the GSM radio channel is not utilized. The complexity and the number of the messages in the EAP-SIM exchange makes it unrealistic for the user to perform this exchange manually without using Bluetooth. Bluetooth security plays a crucial role in this scheme. By compromising the Bluetooth security, an attacker might trick the ME to communicate with attacker's computer and authenticate the attacker.

The SIM Strong authentication that utilizes the SMS channel must also have a mechanism to ascertain that the same user controls both the computer and the ME. This is done with sessionIDs. A sessionID is generated by the dedicated applet residing on the SIM card [99]. The generated sessionID is input by the user in the computer and transferred via Internet channel to the AS. Then the applet performs mutual EAP-SIM authentication with the AS over the SMS channel. The benefit is that this scheme does not require Bluetooth connection [100].

Both types of the SIM Strong authentication require specialized applet on the SIM card to perform EAP-SIM [100].

EAP-AKA

EAP-AKA, defined in RFC 5448, is a mechanism for authentication and session key distribution that uses an UMTS Authentication and Key Agreement (AKA) mechanism. Unlike EAP-SIM that is based on GSM AKA, this authentication protocol is designed to work in 3rd generation networks. EAP-SIM and EAP-AKA were developed in parallel; hence, these protocols have many common ideas [102].

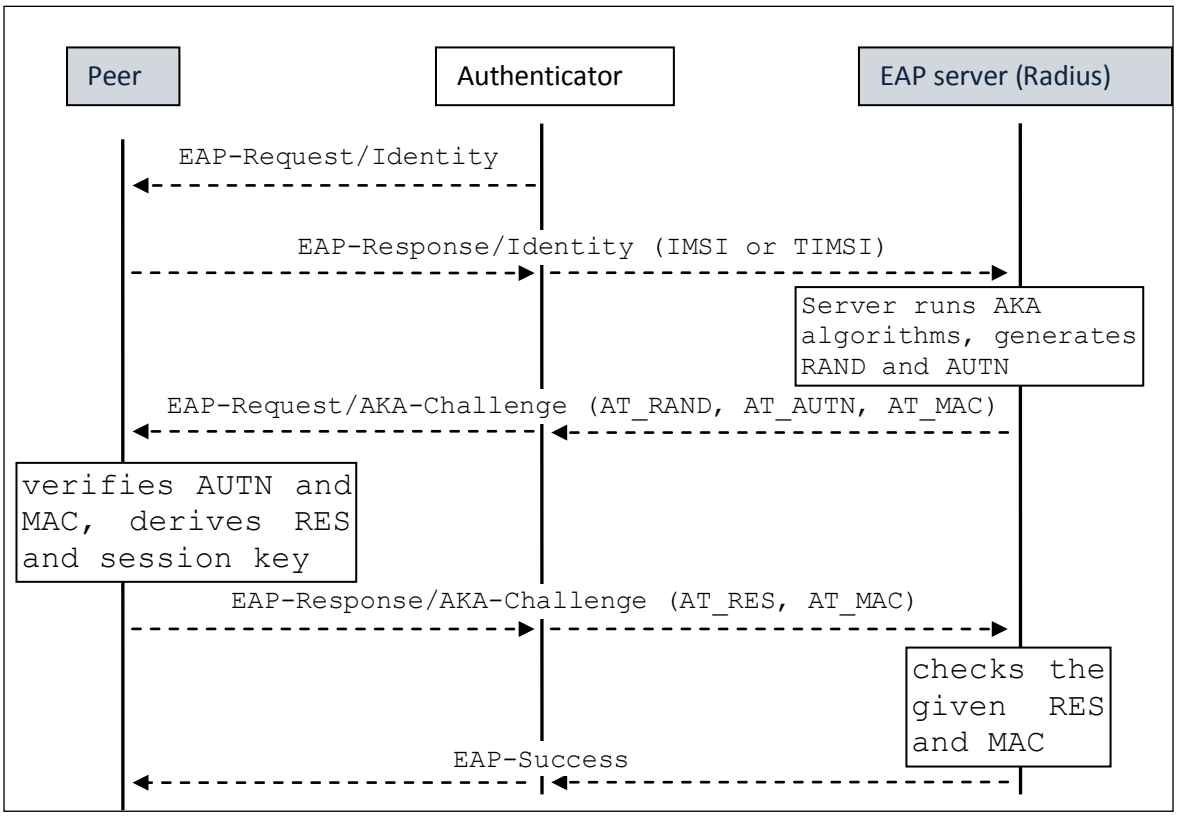

Figure 3.3.2.2: EAP-AKA authentication

Like EAP-SIM, EAP-AKA provides mutual authentication, integrity protection, key derivation, confidentiality protection, and optional identity privacy protection. EAP-AKA can be used for SIM Strong authentication in the same way as EAP-SIM.

3.3.3 One Time Password schemes

One-Time Password schemes are one of the simplest and most popular forms of two-factor authentication [104]. They are designed to be immune to the replay attack. Although the Public-Key Infrastructure (PKI) and the biometric authentication are considered to be stronger than OTP, OTP schemes still provide considerable protection, better than ordinary passwords.

OTP values should be generated in an unpredictable way, so that it would be impossible to derive a new values based on a set of known OTP values. OTP schemes can be classified according to the OTP value generation method:

- **S/KEY like methods**: this method generates a limited set of one-time passwords by performing multiple hashing operations on the initial secret value. The security of this scheme is based on the non-reversibility of the hash function.
- **Counter-based:** the scheme is based on the ever-increasing counter and a secret shared by the token and the AS. RFC 4226 describes a counter-based scheme that uses HMAC-SHA-1 function to generate OTP value. HOTP(K, C) = Truncate(HMAC-SHA-1(K,C)), where K is a shared secret key and C is a counter value [104]. Truncation is used to enable the user to easily enter the resulting value in computer. It is important for the counter to be synchronized between the token used by the user and the authentication server.
- **Time-based**: The usage of time as an input parameter to the OTP generator ensures that the OTP values do not repeat. As described in [105], the time-based OTP (TOTP) value can be generated by hashing the shared secret and a time parameter. Time synchronization is crucial for the TOTP scheme.
- **Challenge-based**: The OTP value is generated based on the shared secret and the nonce received from the verifier. The verifier generates a random value and sends it to the claimant. OATH Challenge-Response Algorithms, described in [106], provides one-way or mutual authentication based on the challenge-response concept. The advantage of this scheme is that parties do not need to maintain synchronization of any values.

OTP from PC to SMS

This scheme is a multi-channel challenge-based OTP authentication system. It is a one-way authentication scheme. Only the user is authenticated to the service provider. Figure 3.3.3.1 depicts the authentication exchange.

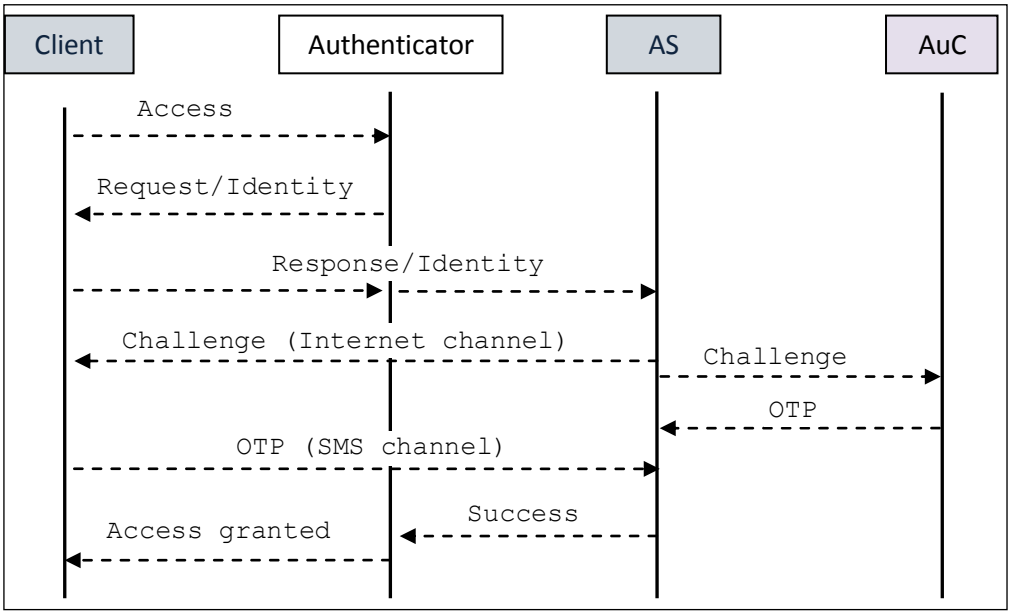

Figure 3.3.3.1: OTP from PC to phone authentication [99]

When the client receives a challenge from the AS via the Internet channel he can either enter this value to the ME manually, or a Bluetooth connection between the computer and the ME can be used to transfer the challenge. The Java MIDlet on the mobile phone communicates the challenge to the applet on the SIM card. The applet takes the challenge as an input, generates an OTP value, and sends it to the MIDlet. Then, either the user manually creates an SMS with generated OTP and sends it to the AS, or the MIDlet can do these steps automatically.

The OTP PC to SMS scheme requires a dedicated OTP generation application on the SIM card. By sending the OTP value via GSM/UMTS SMS channel the user assures AS that he controls both the computer and the ME. The security is based on the assumption that only the legitimate subscriber's SIM can generate OTP and on the fact that the radio channel is encrypted, making it almost impossible to intercept SMS and extract OTP value.

This scheme provides neither confidentiality protection nor temporal key derivation. Besides, since this scheme does not provide mutual authentication and integrity protection it is susceptible to many attacks. As in SMS authentication with sessionID check scheme, it is also possible to launch session hijacking attack. This attack can be easily executed if the attacker controls one of the intermediate nodes between the user's computer and the service provider. When the user starts authentication to the server provider the attacker also starts authentication (supplying the identity of the victim) to the same or the other service provider that uses services of the same AS. The attacker blocks the original request from the user so that it does not reach intended service provider, and responds with a forged Request/identity to the user. When the attacker receives a challenge via Internet channel from the AS, he "forwards" it to the victim. The user will receive the challenge from the attacker. The computer/user/ME has no way to check that the received challenge was actually issued by the AS to provide authentication for the user's session. Believing that this challenge was intended for him the user will unblock OTP generation function by supplying the PIN to the ME, the OTP value will be generated and sent to AS. AS will check whether the received and self-generated OTP values match and will authenticate attacker.

If a Bluetooth connection is used between the ME and the computer, it should be well protected. The attacker can authenticate to any service provider registered at the AS even without the user noticing it (unless the PIN is required to issue confirmation message), if the

Bluetooth security is compromised.

OTP from SMS to PC

Like the architecture with session-IDs, this architecture is based on the fact that a user is already authenticated with the GSM/UMTS network. Therefore, the authentication process consists of the steps that ascertain that the owner of the ME is the same user that controls the computer.

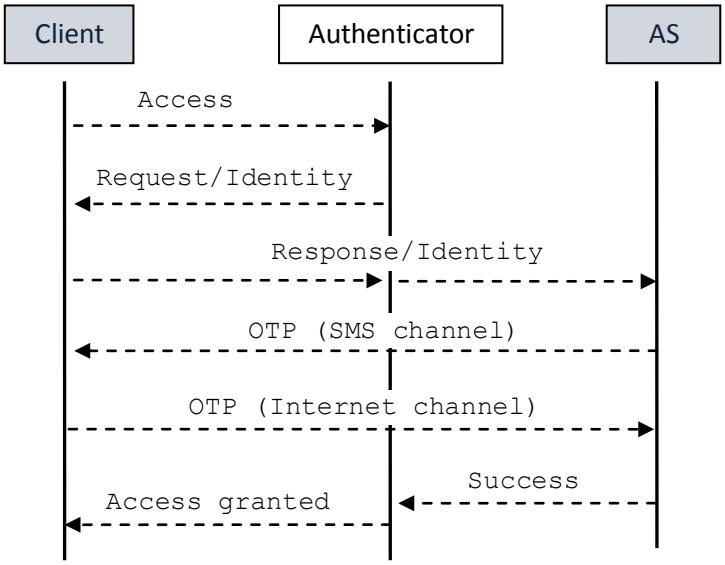

Figure 3.3.3.2: OTP from SMS to PC authentication [99]

Since the AS does the check, the user does not have to verify that session-IDs match as in the architecture with session-IDs.

The Authenticator redirects the session to the AS. The AS generates an OTP based on the user's identity by using a hash function. When the user receives an SMS with the OTP from the AS, the user enters the OTP value into the browser (can be done automatically via Bluetooth). Then the AS verifies whether the received value matches the sent value, authenticates the user, and redirects the browser back to the Authenticator.

The automatic version of this scheme utilizes a Java applet on the computer to get the OTP from the SIM via SAP.

The OTP SMS to PC scheme utilizes the fact that the user is already authenticated in the GSM/UMTS network. Thus, the AS needs to confirm that the owner of the ME actually controls the computer. This is done with an OTP exchange. Although the OTP value is sent to the user's ME via the SMS channel, it is not used to provide mutual authentication. The only difference between the randomly generated sessionID that could be used and the OTP value is that the latest is actually associated with the HTTP session created by the user [99]. To do session hijacking the attacker needs to send OTP value from his computer. However, the attacker cannot intercept this value on the radio channel since this link is protected by GSM/UMTS security mechanisms. Only if Bluetooth is used between the ME and the computer can the attacker obtain OTP value by compromising the Bluetooth security. Though there is no need for the attacker to do this. By intercepting the OTP value, sent back to the AS via Internet channel, and sending it from his computer attacker can hijack the session. Since the OTP value is bound to the HTTP session, the attacker cannot authenticate to arbitrary service provider if the user's computer performs OTP check.

If Bluetooth connection is used between the ME and the computer, it should be well protected. The attacker can authenticate to any service provider registered at the AS even

without the user noticing it (unless the PIN is not asked to authorize the transfer of OTP value to the computer), if the Bluetooth security is compromised.

Session hijacking is possible since this scheme does not provide integrity protection. It also does not provide confidentiality protection and temporal key derivation mechanism.

Enhanced OTP from SMS to PC authentication

This architecture, described in [103], is an enhanced version of the multi-channel challenge-based OTP from SMS to PC solution, which provides integrity protection.

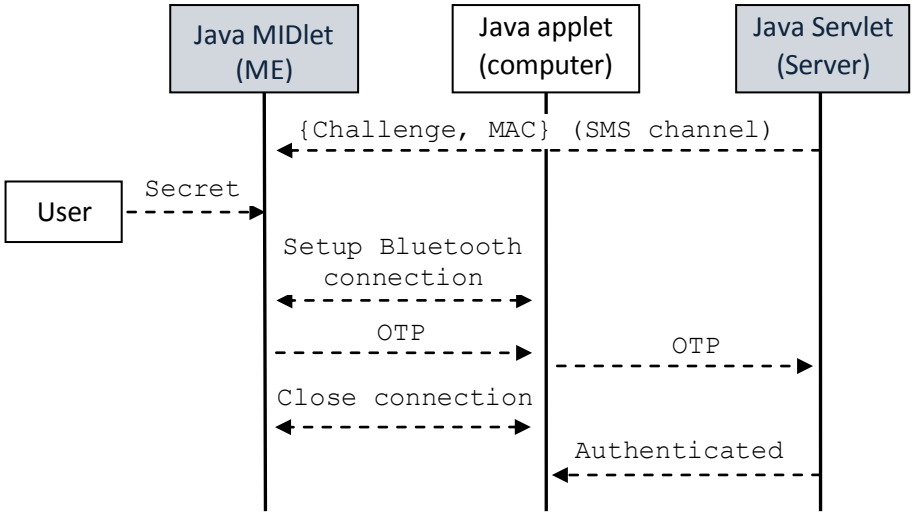

Figure 3.3.3.3: Enhanced OTP based SMS to PC authentication [103]

When a user wants to log on to the server, he presents his user name to the Java Servlet present on the server. The server generates a challenge and computes an OTP value: OTP = hash (challenge || secret key). The message authentication code (MAC) is generated over the OTP. Then the challenge and the MAC are sent to the MIDlet via the SMS channel. Upon receival of the SMS, the MIDlet in the ME is automatically activated and asks to enter the password. The usage of a wireless Messaging API (WMA) enables the MIDlet in the ME to send and receive SMS messages. When an application management software (AMS) on the mobile phone receives an SMS on the port that the MIDlet is registered with, it delivers it to the MIDlet. When the user enters the password, it is hashed and then compared with the stored value. If the check succeeds, the password is used to the decrypt the secret key. Then the MIDlet generates an OTP value by hashing the received challenge and the secret key. The MAC is generated over the OTP and matched with the MAC received via the SMS message. If it is different the procedure is aborted. If they match, the MIDlet starts a Bluetooth connection with the Java applet on the computer and communicates the OTP value. The OTP value may be entered in the computer manually without the usage of Bluetooth, though it should be truncated for usability. The applet sends the OTP to the Java servlet via the Internet channel, and the Bluetooth connection between the Java MIDlet and the Java applet is aborted. The server compares the sent OTP with received OTP and authenticates the user.

The Java MIDlet allows the mobile phone to act as a token. After the user registers at the AS through the Internet, he is able to download the MIDlet. Download is done automatically (user's agreement is required) when the registration is finished. After the MIDlet is installed, the key exchange is started between the MIDlet and the AS via the SMS channel. The aim of this key exchange is to derive a shared secret key used later for OTP generation. A Simple Password Exponential Key Exchange (SPEKE) protocol, which is an improved version of Diffie-Hellman, is used for the key exchange [103].

The server generates an initial OTP and shows it on the web page when the Java MIDlet is downloaded. This initial OTP is used during the key exchange to authenticate the key exchange [103]. Then a user selects a PIN code for the MIDlet. It is used to provide a two-factor authentication.

The Java applet is used to automate the transfer of the OTP from the MIDlet to the computer. It registers a Bluetooth server and waits for connections. A 16-byte pass phrase is used to protect the Bluetooth pairing. The applet passes received OTP values to the server.

The Java servlet represents the AS to the clients. When a user downloads the MIDlet the AS initiates the key exchange. The key exchange is performed over the SMS channel. To send and receive SMS messages, the servlet uses an SMS gateway. The shared secret key derived during the exchange is stored in the AS (along with the user's profile) and in the mobile phone. In the mobile phone, it is stored in the J2ME Record store in the encrypted form. It is encrypted with a password that the user selected. In the MIDlet a hash of the user's password is stored. Therefore, the MIDlet can verify the correctness of the password entered by a user.

It is important to note that the user does not need to prove his identity, though he needs to prove that he controls both the computer and the ME. The security relies on the fact that the ME with subscriber's UICC card is already authenticated by the mobile network operator. The OTP value that is used as the input to the hash function ensures that the user, with whom the key exchange via SMS is performed, is the same user that requested the service from the server (it means that the user controls both the computer and the ME). Only the legitimate subscriber could receive SMS messages, decrypt them, and compute shared secret K based on the initial OTP value and the values sent in SMS. The GSM/UMTS network provides encryption of the radio channel and ensures that SMS is forwarded to the destined receiver.

In the subsequent exchanges the user proves that he controls the computer by sending the OTP value, generated by the ME based on the shared secret and the challenge sent via SMS, via the Internet channel to the AS. Along with the challenge the MAC value (computed on OTP) is sent to the user via the SMS channel. After computing the OTP value, the user can check whether the challenge was sent by the party that knows the shared secret key derived during the key exchange phase. Only the party that knows shared secret could compute OTP value and calculate the MAC value of the OTP. Thus, the user is ensured that the challenge is sent by the same server that displayed the initial OTP value on the web page. However, it is still not the mutual authentication, since the user does not check the identity of the server when making initial HTTP request to the server, thus masquerade attack is possible. The attacker can intercept the initial HTTP request, respond with the initial OTP value in the HTTP response to the user, go through the key exchange phase and derive the shared secret for OTP generation. This scheme provides only integrity protection. The initial OTP value cannot be used to authenticate the server (even if it were based on the secret shared by the server and the user) since it is transmitted openly to the user, and the attacker can intercept it.

Since the shared secret for the OTP generation is dynamically derived by the Java MIDlet, it is not mandatory for the MNO to act as IDM provider. The server can execute SMS exchange with the user directly. The advantage of this scheme is that the dedicated applet on the UICC card is not required. However, the shared secret key used for the OTP generation has to be stored in the encrypted form in the Java Record Store.

3.3.4 Summary

EAP-SIM/EAP-AKA based authentication schemes described in this chapter provide the strongest security level. They provide mutual authentication, integrity, confidentiality, replay protection, and temporal key derivation. These schemes require special applets on the UICC card. The rest of the described schemes provide only one-way authentication.

The enhanced OTP from PC to SMS authentication provides also integrity protection. Besides, this scheme is quite flexible. It is very easy to change IDM provider, since there is no need for a dedicated applet on the UICC card, OTP generation is done in the ME by the Java MIDlet.

It is possible to enhance all authentication schemes so that they would provide mutual authentication and integrity protection. However, this would require subsequent security review of the modified schemas by security experts. Thus, EAP-SIM/EAP-AKA based schemas look more attractive in this context.

Both, the SIM Strong authentication based on the EAP-SIM that utilizes SMS channel with sessionIDs and the SIM Strong with Bluetooth + Internet channel provide adequate security level. In the SIM Strong with Bluetooth + Internet authentication scheme it would be unrealistic (from the usability point of view) to perform EAP-SIM exchange without Bluetooth. Bluetooth security plays a crucial role in this scheme. Secure dynamic Bluetooth link establishment between the ME and the computer can be achieved by using Bluetooth Secure Simple Pairing (SSP) with Numeric comparison or Passkey entry association models to confirm/authenticate peering. This would allow to avoid a challenging management of PINs used in legacy Bluetooth peering schemes.

4. Proposed ME-based logon architectures

In this chapter, proposed mobile phone based logon architectures are described. Both of these architectures rely on the ability of smart cards to securely store credentials. The first proposed architecture corresponds to the 4th extendibility scenario (described in the previous chapter) when neither a custom credential provider nor a custom authentication package needs to be implemented. Consequently, we achieve a seamless integration of the proposed solution with Windows. The second ME-based logon architecture relies on one-time passwords. It corresponds to the third extendibility scenario when an existing credential provider is sufficient, but existing authentication packages do not support required authentication algorithm, thus a custom authentication package needs to be implemented.

4.1 Bluetooth smart card reader architecture

In this architecture, a mobile phone with operator's UICC emulates a PIV- or an IDMP-compatible smart card that interacts with a PC/SC smart card reader residing in the computer via protected Bluetooth channel. The UICC card contains a Java Card applet that manages a public-private key pair. This public-private key pair is used for authentication to the domain controller. The protocol for authentication is the Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). Therefore, a Kerberos infrastructure and an established Public Key Infrastructure (PKI) are required to be present in the domain.

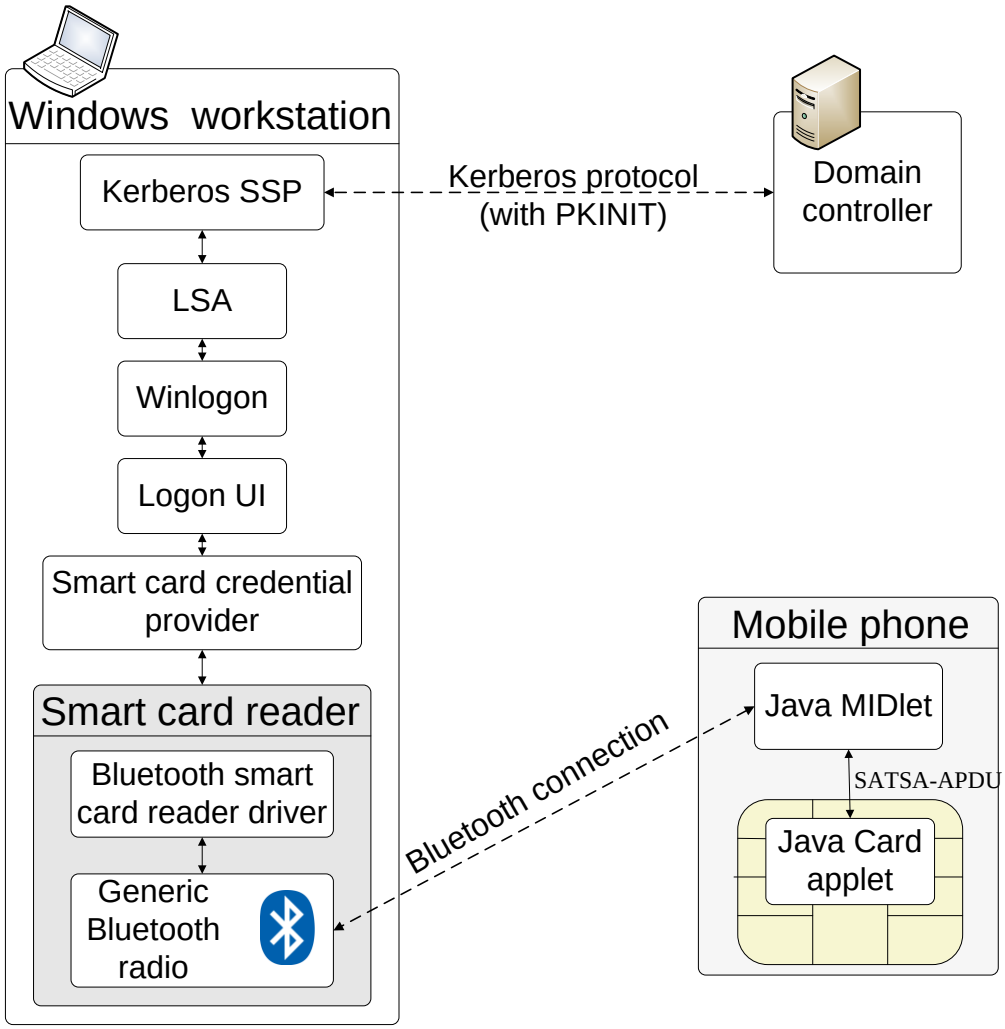

Figure 4.1.1: ME-based logon architecture with a Bluetooth smart card reader server

The main feature of the proposed architecture is that it can support virtually any

authentication scheme that utilizes smart cards. It can be used not only for the domain logon, but also for other purposes like signing documents, e-mails, etc.

The Windows inbox smart card mini-driver is automatically used in case a mobile phone emulates a PIV- or IDMP compatible card. Another alternative would be to create a full proprietary smart card mini-driver specially designed for our virtual phone-based card instead of emulating PIV or IDMP cards.

None of the Windows smart card framework elements is changed. A Bluetooth device (inbuilt or externally connected) together with the developed Bluetooth smart card reader driver acts as an ordinary smart card reader. The mobile phone emulates a PIV-compatible smart card. The default Windows smart card credential provider is responsible for collecting credentials from the emulated smart card and a PIN from a user; authentication is handled by the Kerberos authentication package. The PIN is used to access operations involving the private key, because the Java Card applet residing in the UICC in the mobile phone requires a PIN code to perform these operations.

The Bluetooth connection between the mobile phone and the computer is encrypted and authenticated.

4.1.1 PKINIT

The PKINIT defines protocol extensions to the Kerberos protocol specifications that integrate a public key cryptography into the initial authentication exchange [107]. Addition of the public-key cryptography to Kerberos eliminates users' burden of managing strong passwords. The client and the KDC use public and private key pairs to mutually authenticate during the Client – Authentication server exchange and to derive the encryption key to encrypt the AS_REPLY.

The PKINIT exchange consists of the following steps [107]:

1. The client includes a pre-authentication data element in the initial request to indicate the usage of the public-key authentication. This pre-authentication data element contains the client's public-key data and a signature.
2. The KDC validates the client's X.509 certificate and uses the client's public key to verify the signature. If the client's request is valid, the KDC sends a usual AS_REPLY, but the reply is encrypted using one of the following:
    - a key generated through a Diffie-Hellman (DH) key exchange with the client, signed using the KDC's signature key
    - a symmetric encryption key, signed using the KDC's signature key and encrypted using the client's public key.
3. The KDC's reply contains a pre-authentication field with keying material required by the client to obtain the encryption key for decrypting the encrypted reply fields.
4. The client validates the KDC's X.509 certificate and the KDC's signature, obtains the encryption key, decrypts the reply, and then proceeds in accordance with the ordinary Kerberos protocol.

4.1.2 Description of components

This section provides a description of implemented components. UML diagrams and source code for some of the components are in the Appendix A. The source code for the rest of components of this architecture is in the Appendix C: digital attachments.

Bluetooth smart card reader driver

This driver, called BthScrdSrv.sys, is a kernel-mode function driver that acts as a Bluetooth L2CAP server (server-side Bluetooth profile driver) and at the same time as an ordinary smart card reader driver. It was developed based on two samples from the Windows Driver Kit 7600: BthEcho and Pscr. The BthScrdSrv.sys driver is a key element of the architecture. The driver declares itself as a member of the smart card reader device group, and it advertises its L2CAP-based services via SDP.

During the installation, the system is searched for the first available local Bluetooth radio. If an appropriate device is found, the installation of a local service is triggered.

As every profile driver, BthScrdSrv uses Bluetooth request blocks (BRBs) to send requests to the Bluetooth driver stack. To receive connection notifications from remote devices and to be able to build and send BRB requests to accept incoming connections, the driver registers a Protocol/Service Multiplexer that L2CAP Bluetooth devices connect to. Then it registers itself as a server capable of receiving L2CAP connections. During the L2CAP server registration, the driver specifies a callback function that the Bluetooth driver stack calls to notify the driver about incoming L2CAP connections. Then it creates a SDP record and adds it to the local SDP server. This SDP record contains a list of service classes of which the server is an instance. In our case, a unique 128-bit UUID value is used to define it as an instance of the "Bluetooth smart card reader" service class, and the pre-allocated UUID-16 0x0100 defines it as an L2CAP service. The Java MIDlet in the mobile phone creates a service search pattern based on these two UUIDs to find our server.

The BthScrdSrv registers with the Smart Card Driver Library (Smclib.h). This library standardizes most of the smart card reader driver's functions. It processes most of the I/O control (IOCTL) requests that the resource manager sends to the smart card reader driver [108]. The driver's DeviceControl routine passes requests received from the system to the smart card library. The smart card library automatically completes the calls that do not require interaction with the card. If, for example, the resource manager sends the IOCTL_SMARTCARD_TRANSMIT request, then this request is forwarded to the smart card library that realizes that this request cannot be completed automatically without driver's intervention. Therefore, it calls the driver's RDF_TRANSMIT callback function that the driver registered with the library. This function creates a request for data transmission and sends this request to the Bluetooth driver stack, which in turn sends data to the mobile phone. The BthScrdSrv's Bluetooth transmit and receive functions work asynchronously, however the Smclib expects synchronous response from the RDF_TRANSMIT callback function. Therefore, driver's transmit function waits for a certain period of time (500 ms) for the response from the mobile phone, and if the response does not arrive in time, it responds with a STATUS_IO_TIMEOUT to the Smclib. If the answer was received before time-out, the driver's transmit function sends the mobile phone's response to the Smclib. The smart card library completes the IOCTL_SMARTCARD_TRANSMIT request and sends the mobile phone's response to the system.

When the mobile phone establishes an L2CAP connection with the BthScrdSrv, the driver notifies the system that "the card" is inserted. This event makes the system search for an appropriate mini-driver. The system sends the APDU command to SELECT the PIV AID. The BthScrdSrv transfers this command and any subsequent commands without any modification to the mobile phone that emulates the card. When the driver receives a response from the mobile phone, it sends it back to the system.

Java MIDlet L2CAPClient

This module is responsible for the whole "card emulation logic", for managing the Bluetooth connection with the Bluetooth smart card reader server, and for providing a GUI to

the user. The L2CAPClient works as a communicator between the Windows system and the Java Card applet. It processes and automatically answers requests that do not require involvement of the Java Card applet. Since the Java Card applet understands only a limited set of specially crafted commands, the L2CAPClient acts as an interpreter. It transforms commands received from the BthScrdSrv to the form acceptable by the applet and vice versa. The L2CAPClient examines the received APDU command, and if the Java Card applet's intervention is required, for example to get the public key or to sign a nonce, then it constructs a special APDU command (the one that the Java Card applet knows how to process) and sends it to the applet. After the response is received from the applet, the L2CAPClient MIDlet constructs an APDU response that Windows expects to receive.

After the Windows system shows the logon screen, the user has plenty of time to launch the L2CAPClient MIDlet and establish a Bluetooth connection, in other words "insert the card". If the user presses SAS before establishing the connection, the smart credential provider will tell the Logon UI to show a tile "Insert a card". When the user launches the MIDlet, the MIDlet starts searching for services advertized by the BthScrdSrv. When a device advertizing these services is found, the L2CAPClient establishes a connection to the Bluetooth smart card reader server and waits for the requests. The communication between them conforms to the request-response paradigm when the server sends requests and the client satisfies them. The communication channel is encrypted and authenticated.

Java Card applet JCpki

This Java Card applet resides in the UICC card. It is responsible for creation and modification of the public-private key pair. The private key never leaves the applet. All sensitive operations like signing or renewal of the public-private key pair are PIN-protected. It means that the L2CAPClient must first provide a PIN to the applet to unlock the cryptographic functionality. The JCpki applet works with RSA cryptographic algorithms, though it is possible to use the Elliptic Curve Cryptography (ECC). Java Card and Windows support ECC, so if the UICC card supports it (some cards do not support all APIs defined in Java Card specifications), then it can be used for logon.

The JCpki applet does not depend on the 3GPP applets (USIM, ISIM, etc.) that reside on the card. It does not communicate with them. The Java Card inter-applet firewall ensures that there is no interference from other applets.

For the communication between the MIDlet and the JCpki applet the Security and Trust Services API for J2ME (SATSA) - JSR 177 is used. In particular, the SATSA-APDU package that is based on Application Protocol Data Units (APDUs) is used.

Loading and installation of this applet on the UICC should be carried out according to GlobalPlatform specifications. The GlobalPlatform's delegated management with security domains allows an application provider to perform loading, installation, and deletion of its applications. The issuer of the UICC card should give keys to a security domain that has permissions to do these operations.

4.1.3 Comparison with existing ME-based authentication schemes

In this section, security issues of the proposed architecture are discussed. A comparison of the proposed "Bluetooth smart card reader with PKINIT" architecture with the SIM Strong authentication is made, since the SIM Strong scheme is the strongest among all architectures evaluated in the previous chapter.

Security of the proposed scheme relies on:

- Security provided by smart cards, which is based on logical and physical security mechanisms that form a unified system. Smart cards allow storing information

and executing cryptographic protocols in a secure manner. In the context of the proposed scheme, this means that the public-private key pair is stored securely, and the execution environment for the JCpki's algorithms is protected.

- Java Card platform security. Among other things, it provides the transaction atomicity and the inter-applet firewall.
- Security provided by the GlobalPlatform. The GlobalPlatform provides secure delegated application management with its security domains.
- Security provided by Bluetooth. The Bluetooth channel is encrypted and mutually authenticated. However, there is no integrity protection and the E0 stream cipher used for the encryption has some flaws.
- Security provided by the Kerberos protocol with the PKINIT extension. It includes mutual authentication, integrity, confidentiality, and replay protection.

There is one more link that is located in the ME between the L2CAPClient MIDlet and the JCpki applet. It does not have any protection, except the fact that the JCpki applet asks for a PIN code. Any MIDlet that knows a PIN code can communicate with the JCpki. Risk can be mitigated by Java protection domains that determine access to protected functions. A MIDlet acquires permissions to perform sensitive operations through the security domain after it is installed there. However, modern mobile platforms contain not only the Java virtual machine. The threat can come from other applications. Application security is quite important, because we do not want some other application to steal the PIN code and make the JCpki perform operations involving the private key.

In comparison with the SIM Strong authentication, based on the EAP-SIM/EAP-AKA, the proposed architecture has several advantages. One of the major advantages is that not only the mobile network operator (MNO) can be an identity provider. In the SIM Strong authentication only the MNO can be an identity provider, since no one else knows GSM/UMTS secret keys. Our scheme is MNO independent. In fact, the proposed scheme easily integrates in already established enterprise identity management systems where the identity management is performed by a company itself. It is designed in a way that it can seamlessly integrate with the Active Directory environment. Moreover, any IAM solution that utilizes smart cards for logon and authentication can easily use the proposed system, because it is not limited to Kerberos as an authentication protocol. It can work with any authentication scheme that utilizes smart cards.

In the proposed architecture, there is no need for all users in a company to have subscription to only one mobile operator to make the scheme work as in the SIM Strong's case. Users can continue using their own subscriptions, unless the UICC does not support the Java Card technology or the GlobalPlatform application management framework, or the MNO does not allow installing additional applets. A company would need to have a contract with MNO for the UICC applet management.

The SIM Strong authentication requires constantly available external Internet channel towards the MNO. If the external Internet channel goes down, the MNO becomes inaccessible and no authentication exchange can be made. The proposed solution does not have such requirement. It is assumed that the authentication server would be located inside of the company's perimeter, though no limitations are set.

The SIM Strong authentication that utilizes the Internet+Bluetooth channel is quite similar to our architecture in terms of communication channels. Bluetooth security is crucial in both the SIM Strong scheme and in our scheme, and it is well known that it has some flaws. However, a slightly different variant of the proposed scheme that utilizes a USB connection instead of Bluetooth could mitigate many risks (the same is true for the SIM Strong). It is possible to write either a kernel-mode or a user-mode USB smart card reader driver that would

work with mobile phones.

The other SIM Strong scheme that uses sessionIDs and the SMS channel can be used without Bluetooth. However, this scheme requires available radio connection with the mobile network. It could be a problem in some premises (e.g. inside of industry buildings). Besides, if the GSM network is used instead of UMTS, there can be some security issues.

Both the SIM Strong schemes and the "Bluetooth smart card reader with PKINIT" require specialized application on the UICC card. In the SIM Strong solution, this application is responsible for the EAP-SIM/AKA exchange, and in the proposed solution it is responsible for the operations with the public-private key pair. Both schemes are highly secure and provide mutual authentication, confidentiality, integrity, and replay protection on the link between the computer and the authentication server. The Bluetooth connection is identical in both schemes.

To make the Windows logon work with the SIM Strong authentication would require changing several Windows components. Although Windows supports EAP framework, it does not use EAP protocols for user logon. It uses it with other networking components to provide network access protection. None of the authentication packages provided by Microsoft has the EAP-SIM/EAP-AKA support. It means that in order to use the SIM Strong authentication for the logon, one would first need to implement an authentication package that supports the EAP-SIM/EAP-AKA authentication protocol. Besides, a custom credential provider should be implemented to work with the SIM Strong credentials. A credential provider for the SIM Strong authentication that utilizes Internet+Bluetooth channel would be different from the credential provider for the SIM Strong authentication that uses sessionIDs and the SMS channel. For the Bluetooth communication, a Bluetooth server module would be required on the Windows system. It could be implemented as a kernel-mode or a user-mode driver that acts as a Bluetooth L2CAP/RFCOMM server. The SIM Strong authentication that uses sessionIDs and the SMS channel would not need this module, unless Bluetooth would be used to communicate sessionIDs to the computer. Therefore, compared to the proposed solution the SIM Strong integration into the Windows logon would require much more effort.

The proposed solution and the SIM Strong schemes provide approximately the same level of security, but I consider the "Bluetooth smart card reader with PKINIT" architecture to be more flexible because of the reasons discussed above.

4.2 OTP-based logon architecture

A proposed OTP-based authentication scheme provides mutual authentication, confidentiality, integrity, and replay protection relying on the OTP Kerberos protocol [109]. A UICC card contains a dedicated Java Card applet that generates an OTP value. The OTP is generated based on a 256-bit secret key, shared with the authentication server, and a time parameter (a counter and a challenge variants are also possible). The generated OTP value is entered by a user into the computer, where it is used for the Kerberos OTP authentication. The default password credential provider collects credentials entered by the user. However, a custom authentication package that supports OTP Kerberos is required.

The authentication procedure can be described as follows. A user starts the Java MIDlet on the ME and provides a PIN code to unlock the UICC Java applet OTP generator. The MIDlet provides the applet with the time parameter to generate a time-based OTP (TOTP). The generated TOTP value is shown on the ME's screen. The user inputs his user name and password into the computer. The standard password credential provider collects these credentials, packs them for the Kerberos authentication protocol and gives to the Logon UI. Eventually the LSA submits these credentials to the custom OTP Kerberos authentication

package. The computer starts Kerberos authentication with the KDC.

The following figure depicts the TOTP Kerberos mobile phone based logon architecture.

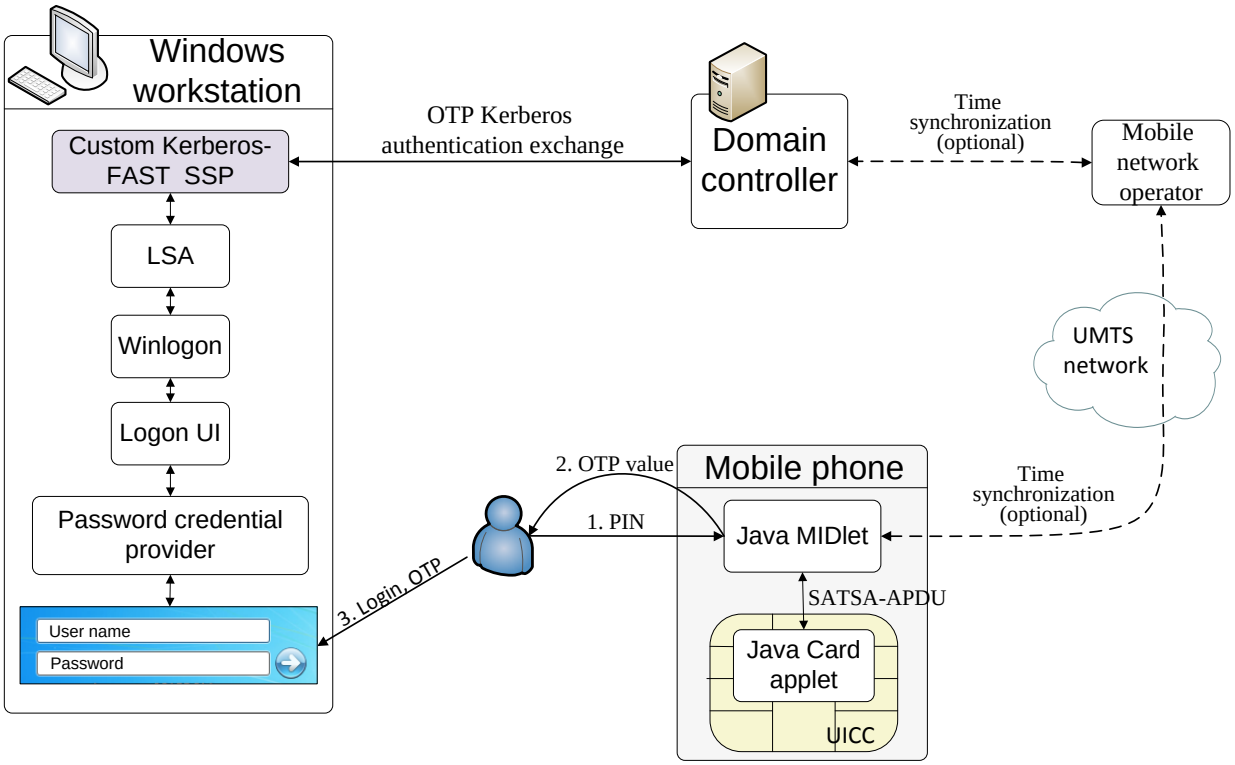

Figure 4.2.1: TOTP Kerberos mobile phone based logon architecture

4.2.1 **OTP Kerberos**

The OTP Kerberos is not a new version of the Kerberos protocol. It is a Kerberos v5 protocol that allows OTP values to be used in the pre-authentication exchange. OTP Kerberos is designed to work with time-based, counter-based, and challenge-response based OTP systems. A Kerberos 4-pass variant is used when the challenge from the server is required, otherwise a 2-pass variant that relies on timestamps/counters is used. In both variants, a client has to generate an OTP response and two secret keys: a Reply Key - to decrypt KDC's reply and a Client key to encrypt messages sent to the KDC.

When a user enters a password, a workstation generates a master key by hashing this password. Then the KDC sends a session key to the workstation. This session key is encrypted with the master key. However, since it is an OTP scheme, we have weak passwords (6 - 8 digit value). Consequently, a simple attack can be used to get the session key by trying all master keys (10^6 – 10^8). Therefore, a secure tunnel is needed. The inner authentication should be bound to the outer secure tunnel in a way that the inner authentication will not be performed without available and working outer tunnel. These all features are already done in the Flexible authentication secure tunneling (FAST) protocol.

The Flexible authentication secure tunneling (FAST) protocol, described in [110], provides a protected channel between the user and the KDC. FAST extends the Kerberos protocol with pre-authentication mechanisms (OTP in our case) by encapsulating the pre-authentication mechanism in FAST messages. FAST does not authenticate the client or the KDC (though FAST allows to optionally authenticate the KDC to the client); it provides a protected tunnel for the pre-authentication data transfer.

FAST uses so-called armoring schemes. The armor scheme must provide a fresh armor key for each conversation. The armor key is used to encrypt the pre-authentication data in the subsequent message exchange. For example, a Ticket Granting Ticket (TGT) obtained by the

user's machine using the host keys to pre-authenticate with the KDC can be used as an armor ticket, or the client can utilize anonymous PKINIT to obtain an anonymous TGT as the armor ticket and optionally authenticate the KDC [110]. These two schemes are ticket-based armor schemes. For these schemes, the armor ticket is used in the derivation of the armor key.

The armor key is used to protect the pre-authentication exchange. It is also used in the derivation of the Client and the Reply keys, in order to bind the outer FAST tunnel to the inner encryption. This is done to mitigate some man-in-the-middle attacks [109].

4-pass system

The client sends an initial AS-REQ. If the OTP authentication is required, the KDC answers with a KRB-ERROR message containing a PA-OTP-CHALLENGE in the FAST protected data. The PA-OTP-CHALLENGE contains a KDC generated nonce and some other optional values. Upon receiving the challenge, the client generates an OTP value and derives the Client and the Reply keys from the Armor Key and the OTP value. A token can generate the OTP value based on the challenge generated by the KDC, token's current state (e.g. time), or a combination of these two [109]. The generated Client Key is used to encrypt the nonce received from the KDC. Then the client generates its own nonce and sends it along with the encrypted nonce (the nonce received from the KDC and encrypted with the Client key) to the KDC in the FAST protected data filed of the AS-REQ message. On receipt, the server generates encryption keys from the OTP value and the armor key, and authenticates the client decrypting the nonce encrypted by the client and checking it. The server responds with the AS-REP that contains the client's nonce encrypted with the Reply key in the FAST protected data.

Figure 4.2.1.1 depicts the 4-pass pre-authentication.

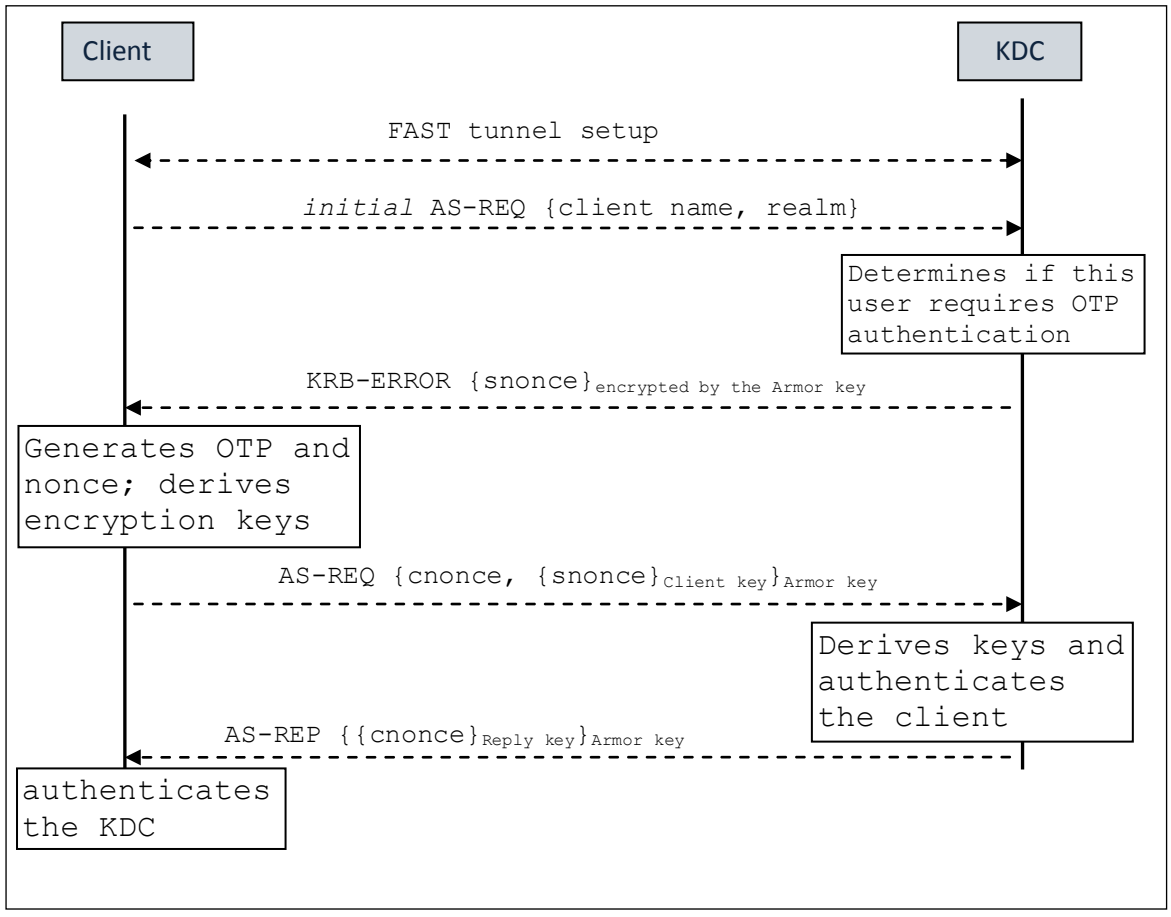

Figure 4.2.1.1: OTP Kerberos 4-pass pre-authentication

The steps involved in the Kerberos 4-pass pre-authentication are the following [103]:

- The user enters the username in the computer/client;
- The client sends the initial AS-REQ with the user's name and realm to the KDC;
- The KDC determines that the OTP authentication is required for the user;
- The KDC replies with the KRB-ERROR message that contains the challenge/nonce. The message is protected by the FAST encryption;
- The computer asks the user to enter the OTP value or connect the token to the computer in order to retrieve the OTP value. The OTP value can be generated based on the challenge generated by the KDC, token's current state (e.g. time), or a combination of these two.
- The generated OTP value and the armor key are used to derive the Client and the Reply encryption keys. The generated Client Key is used to encrypt the s_nonce received from the KDC.
- Then the client generates its own c_nonce and sends it along with the encrypted s_nonce (the nonce received from the KDC and encrypted with the Client key) to the KDC in the FAST protected data filed of the AS-REQ message.
- The KDC generates the OTP value for the user based on the identity from the AS-REQ message. Then the KDC derives encryption keys and authenticates the client decrypting the s_nonce encrypted by the client and checking it.
- The KDC constructs a TGT for the user
- The KDC responds with the AS-REP that contains the TGT and the client's c_nonce encrypted with the Reply key in the FAST protected data.
- The client authenticates the KDC

2-pass system

This scheme can be used if the client knows that the KDC supports the OTP pre-authentication. The client derives the Client and the Reply key from the armor key and the OTP value, and encrypts a timestamp with the derived Client key. Then the client generates the nonce that will be used to authenticate the KDC. Then the client includes the encrypted timestamp and the generated nonce to the PA-OTP-REQUEST (FAST protected) in the initial AS-REQ. On receipt, the server generates encryption keys from the OTP value and the armor key, and authenticates the client decrypting the timestamp encrypted by the client and checking it. The server responds with the AS-REP that contains the client's nonce encrypted with the Reply key in the FAST protected data.

Figure 4.2.1.2 shows the 2-pass pre-authentication exchange.

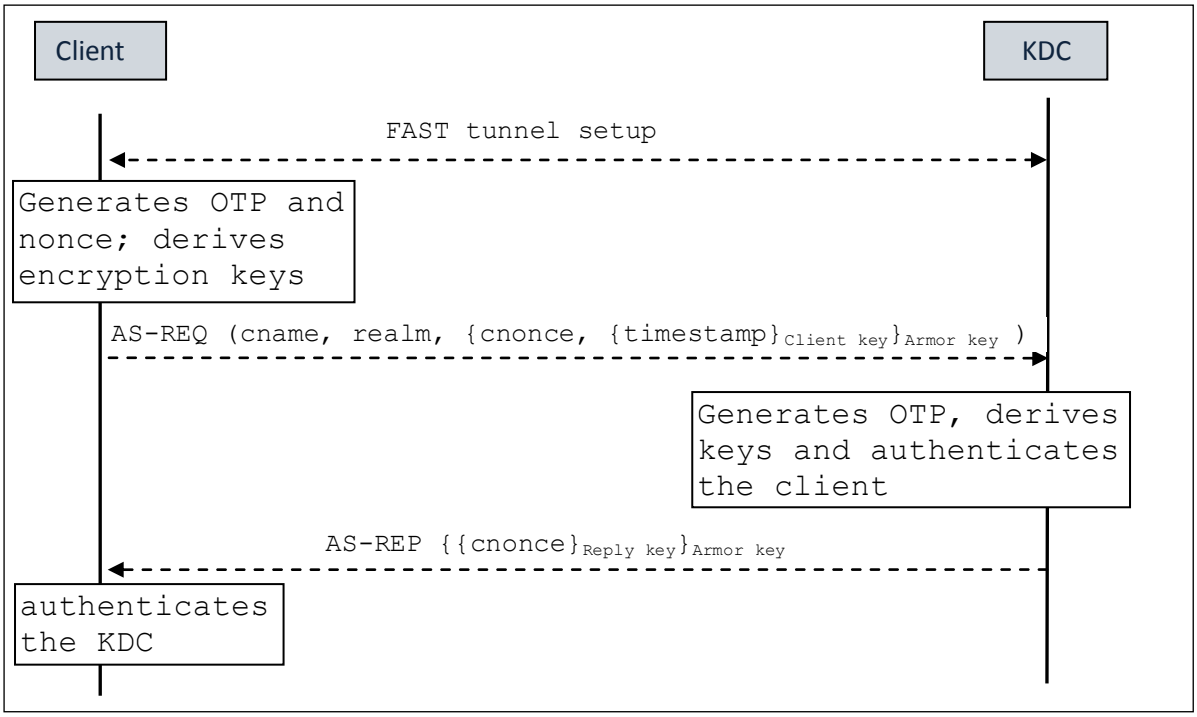

Figure 4.2.1.2: OTP Kerberos 2-pass pre-authentication

In the described 2-pass and 4-pass systems, we assumed that the OTP value is used to derive the Client and the Reply key and is not sent directly to the KDC. However, if the OTP mechanism is a part of the authentication exchange and the OTP value should be send, then the KDC authentication cannot be provided by pre-authentication exchange and should be provided by other means, for example the KDC can be authenticated during the FAST tunnel setup [109].

In the 4-path system, the client is not required to know beforehand which pre-authentication schemes the KDC supports. Besides, in this scheme the KDC, by looking up the clients name and the realm, can determine whether the client requires OTP authentication or the ordinary Kerberos authentication. According to [109], the way it is done is implementation specific. That is why we consider the 4-pass pre-authentication scheme to be more suitable for the mobile phone based logon architecture.

In the 4-pass system, the KDC sends a nonce to the client that can be used for the OTP generation. However, it is much harder to integrate the 4-pass system with OTPs derived from the nonce into the Windows logon architecture than the 4-pass system with OTPs derived from a time value. In the first case, besides the custom Kerberos-FAST authentication package, one would need to write a custom credential provider to communicate the nonce to the ME. Moreover, there are special cryptographic requirements for the length of the nonce. It is not secure to use a short nonce. According to [109], the nonce should be "*as long as the longest key length of the symmetric key types that the KDC supports and MUST be chosen randomly*". Since challenges have considerable length, a user cannot read them from a screen and manually input them into the mobile phone. Therefore, a USB or a Bluetooth connection should be used. And this requires implementing an additional component responsible for this in Windows. That is why the 4-pass system with OTPs derived from time seems to be more appropriate. It can work with the default password credential provider; and a user can simply type in the generated OTP into the computer.

4.2.2 Description of components

This section provides a description of implemented components. UML diagrams and source code for some of the components are in the Appendix B. The source code for the rest of components of this architecture is in the Appendix C: digital attachments.

Java TOTPClient

A mobile phone with a dedicated application on the UICC card acts as a token. The mobile phone contains a Java MIDlet, called TOTPClient, responsible for the authentication logic on the ME and interaction with a user. The TOTPClient simply uses the UICC Java Card applet JCtotp to generate a TOTP.

For the communication between the MIDlet and the Java Card applet the Security and Trust Services API for J2ME (SATSA) - JSR 177 is used. In particular, the SATSA-APDU package that is based on Application Protocol Data Units (APDUs) is used.

JCtotp applet

The UICC Java Card applet, called JCtotp, is responsible for the OTP generation based on a time parameter and a 256-bit secret key shared by the Java applet and the authentication server. The secret key is loaded to the card (along with initial PIN) during applet loading/installation. The shared key never leaves the applet. To generate an OTP value a user would need to unlock the applet's OTP generation function by entering a PIN code. The OTP schema can be a HMAC-Based One-Time Password Algorithm (HOTP) [104] based on counters, or a Time-based One-time Password Algorithm (TOTP) [105]. In this thesis, the TOTP based schema was implemented.

For the TOTP schema to work the mobile phone should synchronize a clock with the authentication server. The MNO could provide a time synchronization service for the authentication server and the mobile phone. The authentication server would actually be able to use any internet time server. Users also could manually synchronize time in their mobile phones with "company's time", though this solution is not quite user friendly.

Since the Bluetooth connection is not used, the generated OTP value is truncated to provide better usability. It is truncated according to the RFC 4226 that describes a counter-based scheme, which uses HMAC-SHA-1 function to generate an OTP value. The time-based OTP value is obtained by replacing the counter with time in this function [105]. The OTP value is generated as: TOTP(K, time) = Truncate(HMAC-SHA-1(K, time)), where K is a shared secret key.

The JCtotp applet does not depend on the 3GPP applets (USIM, ISIM, etc.) that reside on the card. It does not communicate with them. The Java Card inter-applet firewall ensures that there is no interference from other applets.

We assume that the UICC card supports the Java Card technology and the GlobalPlatform application management framework. The Java Card security mechanisms protect the OTP generation applet. The GlobalPlatform allows delegating application management, so that an actual application provider/service provider will be responsible for management of its application and not a mobile network operator that owns the card. Besides, the GlobalPlatform allows the application provider to securely load/change shared secret keys. The issuer of the UICC card should give keys to a security domain that has permissions to do all these operations.

Kerberos-FAST SSP

This custom authentication package should implement a FAST extension to the Kerberos protocol according to the 4-pass scheme with OTP values derived from time.

4.2.3 Comparison with existing ME-based authentication schemes

The security of this scheme is based on the assumption that only a user who possesses the ME with the UICC card, which contains an applet that holds the secret key, is able to generate a valid OTP.

Security of the proposed scheme relies on:

- Security provided by smart cards, which is based on logical and physical security mechanisms that form a unified system. Smart cards allow storing information and executing cryptographic protocols in a secure manner. In the context of the proposed scheme, it means that the public-private key pair is stored securely, and the execution environment for the JCtotp's algorithms is protected.
- Java Card platform security. Among other things, it provides the transaction atomicity and the inter-applet firewall.
- Security provided by the GlobalPlatform. The GlobalPlatform provides secure delegated application management with its security domains.
- Security provided by the Kerberos protocol. It includes mutual authentication, integrity, confidentiality, and replay protection.

All application security considerations for the mobile phone that apply to the Bluetooth smart card reader architecture also apply to this architecture.

In comparison with the SIM Strong authentication based on the EAP-SIM/EAP-AKA the proposed architecture has several advantages. One of the major advantages is that not only the mobile network operator (MNO) can be an identity provider. In the SIM Strong authentication only the MNO can be an identity provider, since no one else knows GSM/UMTS secret keys. Our scheme is MNO independent. There is no need for all users of a company to have subscription to only one mobile operator to make the scheme work as in the SIM Strong's case. Users can continue using their own subscriptions, unless the UICC does not support the Java Card technology or the GlobalPlatform application management framework, or the MNO does not allow installing additional applets. A company would need to have a contract with theMNO for the UICC applet management.

The SIM Strong authentication requires constantly available external Internet channel towards the MNO. If the external Internet channel goes down, the MNO becomes inaccessible and no authentication exchange can be made. The proposed solution does not have such requirement. It is assumed that the authentication server would be located inside of the company's perimeter, though no limitations are set.

Both the SIM Strong scheme and the TOTP Kerberos ME-based scheme require specialized application on the UICC card. In the SIM Strong solution, this application is responsible for the EAP-SIM/AKA exchange, and in the proposed solution it is responsible for the OTP generation.

The SIM Strong authentication that utilizes Internet+Bluetooth cannot work without Bluetooth connection (USB connection could be used), since it is unfeasible to make the EAP-SIM/AKA exchange manually. And it is well-known that Bluetooth has some security issues [60, 98]. On the other hand, the SIM Strong with sessionIDs and the SMS channel can be used without Bluetooth. However, this scheme requires available radio connection with the mobile network. It could be a problem in some premises (e.g. inside of industry buildings).

The proposed TOTP Kerberos ME-based logon architecture provides a high security level, since it relies on the security of the industry standard protocols and technologies such as Kerberos, GlobalPlatform, and Java Card. As an OTP generation algorithm a time-based OTP scheme that is currently an IETF Internet-draft is used. Smart card security mechanisms provide

physical and logical security, thus ensuring protection for the OTP generation algorithm. Since the authentication procedure does not rely on the GSM/UMTS subscription secrets, the proposed architecture provides flexibility by allowing a company to do identity management itself. The widespread usage of the Kerberos protocol would allow an easy transition to the new authentication scheme. Since only the thoroughly studied technologies are used in the proposed architecture, security risks are mitigated.

5. Conclusions

This master's thesis has proposed two novel Windows workstation logon schemes that use a mobile phone with a UICC card for authentication. The proposed schemes allow users registered in the domain to log on to their workstations. The thesis also studies how new phone-based authentication schemes can be integrated into the Windows logon architecture.

5.1 Results and achievements

The first scheme emulates a smart card reader and a smart card in order to interoperate with the Windows smart card framework to provide PKI-based logon. The mobile phone with the UICC card emulates a smart card that communicates with the emulated smart card reader via protected Bluetooth channel. To emulate the Bluetooth smart card reader only a Bluetooth adapter (in-built or externally connected) is required. The Bluetooth connection is encrypted and authenticated. Neither the phone nor the UICC can emulate the smart card on its own. The UICC card is responsible for the secure storage of the public-private key pair and for the secure execution of cryptographic algorithms. This ensures high security of the proposed solution.

The advantages of the Bluetooth smart card reader architecture are as follows. It provides a two-factor authentication: the ownership of the phone and a PIN to access the UICC. Unlike many solutions available on the market, it is not a "soft token". All secrets are stored on the real smart card (UICC) and not on the phone itself. Therefore, the phone is a secure and tamper-resistant token.

The solution reuses available smart card infrastructure as much as possible, both in terms of software and hardware. It requires only a Bluetooth adapter. No change is required to the domain infrastructure. Therefore, a seamless integration with Active Directory and Window server is achieved. Identity management is done with the same tools as for the real smart cards.

This solution can work with any authentication scheme used with real smart cards. Since a smart card reader is emulated, the proposed solution can be used not only for the logon but also for all other functions typically done with smart cards (e.g. signing of documents, e-mails).

The Bluetooth smart card reader architecture is mobile network operator independent. It does not depend on the GSM/UMTS secrets. Users can use their existing subscriptions, if a mobile network operator allows delegated management according to the GlobalPlatform specifications. Besides, the proposed scheme does not need the GSM/UMTS radio network connection as many other schemes that use mobile phones for authentication.

Although the Windows system really sees the Bluetooth adapter as a fully functional smart card reader and the secure Bluetooth communication was achieved between the phone and the reader, because of the lack of time the module responsible for processing commands sent to the emulated card can process only several commands. Therefore, the complete authentication procedure was not achieved. However, it was shown that the idea is working.

In the second scheme, the mobile phone with the UICC card serves as a token for generating OTP values based on a shared secret key and the time parameter. The UICC card is responsible for the secure storage of the shared secret key and for the secure execution of the OTP generation algorithm. This ensures high security of the proposed solution. This thesis concentrates on the first authentication scheme, therefore the second scheme is described in addition. Only the client's part of the OTP-based scheme was developed.

The thesis also studies how new authentication schemes in general and those that work with mobile phones in particular can be integrated into the Windows logon system. A conclusion is made that it is impossible to make a generic architecture that would easily support all existing and possible future mobile phone authentication schemes for the Windows logon.

Windows is already a highly customizable environment and can support virtually any authentication scheme for the logon, though a considerable amount of modifications may be required to implement a particular scheme.

All authentication schemes can be placed to one of four groups according to the amount of modifications required to be done to the Windows system. The evaluation of the difficulty level for each of the groups is made. As an example, the SIM Strong architecture is evaluated according to this scheme. It is shown that the SIM Strong, as well as the rest of evaluated phone-based architectures, requires changing several Windows components (quite a demanding task) in order to be used for the workstation logon.

5.2 Discussions and future work

Both of the proposed architectures rely on the UICC card as a security element. However, since secret keys for the UICC's security domains were not available, the developed Java Card applets were tested only in the development kit simulation environment. There was no possibility to make it with real cards. Therefore, both solutions should be tested with real cards.

The key element of the Bluetooth smart card reader architecture – the driver has been fully implemented. It was tested on Windows 7. Although there should be no problems in using it on Windows Vista or XP, these tests are still to be done. The Bluetooth smart card reader driver securely and properly transmits all received APDUs in both directions, however because of the lack of time the L2CAPClient MIDlet, which is responsible for the card emulation logic, has limited APDU processing capabilities. It can process only simple commands like selecting applet. This drawback does not allow showing a fully functional logon procedure. However, there should not be any difficulties to make this module fully functional.

The main feature of the proposed Bluetooth smart card reader architecture is that it can support virtually any authentication scheme and application that utilizes smart cards. Therefore, in the future work we would like to try it in such widespread operations as signing documents, e-mails, etc.

The proposed Bluetooth smart card reader architecture utilizes Bluetooth for communication between the phone and the computer. However, the Bluetooth encryption algorithm has some security flaws [60], and is not considered very strong [98]. On the other hand, the USB connection offers higher security since it is almost impossible to eavesdrop data transferred over a short USB cable between the mobile phone and the computer. Besides, there is no need for authentication mechanisms, because you see where you plug-in the cable. In addition, USB has higher data transfer speed and does not have radio media problems such as interference. Since Windows already has a standard USB smart card reader driver, only the mobile phone part would need to be changed (though, a smart card mini-driver may also be required). In particular, a module that manages a USB connection should be implemented. Nowadays, not all phones are equipped with USB ports, but in the nearest feature, it will be on every phone because the Micro-USB was accepted as a common universal charging interface by leading phone producers.

This thesis primary concentrates on the Bluetooth smart card reader architecture and its implementation. Therefore, not so much time was devoted to the second proposed architecture. In the second OTP-based proposed architecture only the mobile phone part was implemented (the Java MIDlet and the Java Card applet). There was not enough time to implement the Kerberos-FAST authentication package. There is a possibility to use the IPSec protection instead of the FAST tunnel to provide an encrypted outer tunnel for Kerberos. Then, the default Kerberos authentication package may be used. However, a thorough analysis is required to understand whether this substitution with IPSec will not dramatically decrease the overall

security. The Active Directory does not have native support for one-time passwords. So, some external servers may be required to make OTP schemes work in Windows domains. Therefore, this question requires more careful investigation.

The proposed architectures rely on the ability of smart cards to securely store data and perform cryptographic operations. However, it would be much easier if mobile phone platforms provided similar services. The idea is to study how mobile phone platforms can provide secure storage and protected execution environment without relying on smart cards.

6. References

[1]. Do Van Thanh, Ivar Jorstad, "The Ambiguity of Identity", telektronikk 3/4.07, 2007, available at: <http://www.telenor.com/telektronikk/volumes/pdf/3_4.2007/Page_003-010.pdf>

[2]. H. Noonan, "Identity", Stanford Encyclopedia of Philosophy, 2009, available at: < http://plato.stanford.edu/entries/identity/>

[3]. "Glossary of Key Information Security Terms", NIST IR 7298, 2006, available at: < http://csrc.nist.gov/publications/nistir/NISTIR-7298_Glossary_Key_Infor_Security_Terms.pdf>

[4]. Do Van Thuan, "Identity Management Demystified", telektronikk 3/4.07, 2007, available at: <http://www.telenor.com/telektronikk/volumes/index.php?page=ing&id1=73&id2=200&id3=976&select=05-09>

[5]. "Identity Management Design Guide with IBM Tivoli Identity Manager", 3rd edition, IBM Corp., April 2009, available at: < http://www.redbooks.ibm.com/redbooks/pdfs/sg246996.pdf >

[6]. W. Soyinka, "Linux administration Beginner's guide", 5th edition, McGraw-Hill, 2008

[7]. W. R. Stanek, "Windows Server 2008 insideout", Microsoft Press, 2008

[8]. "Investigating Single Sign-on", Novell white paper, available at: < http://www.novell.com/rc/docrepository/public/37/basedocument.2007-08-07.2321076507/4622014_en.pdf>

[9]. "What is the data store", available at: <http://technet.microsoft.com/en-us/library/cc787905%28WS.10%29.aspx>

[10]. D. Birch, "Digital Identity Management: Perspectives On The Technological, Business and Social Implications", Gower Pub Co, 2007

[11]. C. Neuman, T. Yu, S. Hartman, K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, available at: < http://tools.ietf.org/html/rfc4120>

[12]. M. Benantar, "Access Control Systems: Security, Identity Management and Trust Models", Springer Science+Business Media, 2006.

[13]. "The Role of Kerberos in Modern Information Systems", MIT Kerberos Consortium, 2008, available at: < http://www.kerberos.org/software/rolekerberos.pdf >

[14]. "The MIT Kerberos Administrator's How-to Guide", MIT Kerberos Consortium, 2008, available at: < http://www.kerberos.org/software/adminkerberos.pdf>

[15]. D. Todorov, "Mechanics of user identification and authentication: Fundamentals of Identity Management", Auerbach Publications, 2007

[16]. T. Bialaski, Michael Haines, "LDAP in the Solaris Operating Environment: Deploying Secure Directory", Prentice Hall, 2003

[17]. "Recommended Practices for Deploying & Using Kerberos in Mixed Environments", MIT Kerberos Consortium, 2008, available at: < http://www.kerberos.org/software/mixenvkerberos.pdf >

[18]. S. Huque “LDAP Weaknesses as a Central Authentication System”, University of Pennsylvania, 2007, available at: <http://www.huque.com/~shuque/doc/2007-10-LDAP-Authn.html>

[19]. Jan De Clercq, “Single Sign-On Architectures”, Springer Berlin, 2002, available at: < http://www.springerlink.com/content/806c0atpq9ab0nx4/ >

[20]. Dao Van Tran, Pål Løkstad, Do Van Thanh, ”Identity Federation in a Multi Circle-of-Trust Constellation”, Telektronikk, Telenor, 2007, Volume 103, pp. 103-117

[21]. Do Van Thuan, “Identity Management Demystified”, Telektronikk, Telenor, 2007, Volume 103, pp. 11-18

[22]. ”SunOpenSSO Enterprise 8.0 TechnicalOverview”, SunMicrosystems, March 2009, available at: < http://dlc.sun.com/pdf/820-3740/820-3740.pdf >

[23]. ” Security Assertion Markup Language (SAML) V2.0 Technical Overview”, OASIS Committee Draft, 2008, available at: <http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf>

[24]. “Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS standard, 2005, available at: <http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf>

[25]. R. Harrison, “Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms”, RFC 4513, June 2006, available at: < http://tools.ietf.org/html/rfc4513 >

[26]. “Sun Java SystemDirectory Server Enterprise Edition 6.3 Evaluation Guide”, SunMicrosystems, 2008, available at: < http://dlc.sun.com/pdf/820-2766/820-2766.pdf >

[27]. H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, V. Niemi, “UMTS Networks: Architecture, Mobility and Services”, 2nd edition, Wiley, February 2005.

[28]. “Digital cellular telecommunications system (Phase 2+); Numbering, addressing and identification”, ETSI EN 300 927 V5.4.0, 2000, available at: http://www.etsi.org/deliver/etsi_en/300900_300999/300927/05.04.00_40/en_300927v050400o.pdf

[29]. V. Niemi, K. Nyberg, “UMTS Security”, Wiley, 2003.

[30]. “MSISDN”, Wikipedia, [cited] Mai 2010, available at: <http://en.wikipedia.org/wiki/MSISDN>

[31]. K. Mayes, K. Markantonakis, “Smart cards, Tokens, Security and Applications”, Springer Science+Business Media, 2008.

[32]. B. Holcombe, “Government smart card handbook”, available at: <http://www.smartcard.gov/information/smartcardhandbook.pdf>.

[33]. W. Rankl, W. Effing, “Smart Card Handbook”, 3rd Edition, Wiley, November 2003.

[34]. “MF1ICS70 Functional specification”, Rev. 4.1, January 2008, available at: < http://www.nxp.com/acrobat_download/other/identification/M043541_MF1ICS70_Fspec_rev4_1.pdf>.

[35]. “What Makes a Smart Card Secure?”, Smart Card Alliance Contactless and Mobile Payments Council White Paper, October 2008, available at: < http://www.smartcardalliance.org/pages/download>.

[36]. J. Aussel, “Smart Cards and Digital Identity”, Telektronikk, Telenor, 2007, Volume 103, pp. 66-79.

[37]. “Common Criteria for Information Technology Security Evaluation”, v.3.1, July 2009, available at: < http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf>

[38]. F. Vater, S. Peter, P. Langendorfer, “Combinatorial Logic Circuitry as Means to Protect Low Cost Devices Against Side Channel Attacks”, Springer Berlin, 2007, available at: <http://www.springerlink.com/content/78k77t2m75731737/fulltext.pdf>.

[39]. “Runtime Environment Specification: Java Card Platform, Version 3.0 Classic Edition”, Sun Microsystems, 2008, available at:<http://java.sun.com/javacard/3.0/specs.jsp>.

[40]. “Java Card Platform Security”, Technical White Paper, Sun Microsystems, available at: <http://java.sun.com/javacard/reference/docs/JavaCardSecurityWhitePaper.pdf>.

[41]. “Java Card Protection Profile Collection”, Sun Microsystems, 2006, available at: <http://java.sun.com/javacard/pp.html>.

[42]. “The Java Card 3 Platform”, White paper, Sun Microsystems, 2008, available at: <http://java.sun.com/javacard/3.0/javacard3_whitepaper.pdf>.

[43]. "Overview", GlobalPlatform, 2004, available at: <http://www.globalplatform.org/pdf/GP_Overview_June2004.pdf>.

[44]. "Frequently Asked Questions", GlobalPlatform, available at: <http://www.globalplatform.org/mediafaq.asp#10>.

[45]. "MULTOS – the High Security Smart Card OS", Whitepaper, Multos, 2005, available at: <http://www.multos.com/downloads/marketing/Whitepaper_MULTOS_Security.pdf>.

[46]. "EMV Migration Economics - Comparing Native and MULTOS smart card choices", Multos, available at: <http://www.multos.com/downloads/marketing/Whitepaper_EMV_Migration_Economics_Platform_Comparison.pdf>.

[47]. "Technology", Multos, 2009, available at: <http://www.multos.com/technology/>.

[48]. “UICC-terminal interface; Physical and logical characteristics”, 3GPP TS 31.101 v 8.0.0, 3GPP Organizational Partners, 2009.

[49]. “ETSI TS 102 221 v8.2.0”, ETSI, 2009, available at: <http://www.etsi.org/deliver/etsi_ts/102200_102299/102221/08.02.00_60/ts_102221v080200p.pdf>.

[50]. “Universal Subscriber Identity Module (USIM) application”, 3GPP TS 31.102 v9.0.0, 3GPP Organizational Partners, 2009.

[51]. “IP Multimedia Services Identity Module (ISIM)”, White paper, Giesecke & Devrient GmbH, 2006.

[52]. “IP Multimedia Services Identity Module (ISIM) application”, 3GPP TS 31.103 v8.1.0, 3GPP Organizational Partners, 2009.

[53]. “Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)", 3GPP TS 31.111 v8.7.0, 3GPP Organizational Partners, 2009

[54]. "The Security and Trust Services API for J2ME, Part 1", Sun Microsystems, 2005, available at: <http://developers.sun.com/mobility/apis/articles/satsa1>.

[55]. “Bluetooth Specification v2.1 + EDR”, Bluetooth SIG, July 2007, available at: <http://www.bluetooth.com/Specification%20Documents/Core_V21__EDR.zip>.

[56]. “Bluetooth Specification v3.0 + HS”, Bluetooth SIG, April 2009, available at: <http://www.bluetooth.com/Specification%20Documents/Core_V30__HS.zip>.

[57]. “Basics”, Bluetooth SIG, available at: <http://www.bluetooth.com/English/Technology/Pages/Basics.aspx>, last visited Mai 2010.

[58]. “RFCOMM”, Bluetooth SIG, available at: <http://www.bluetooth.com/English/Technology/Works/pages/rfcomm_1.aspx>, last visited Mai 2010.

[59]. “Object Exchange (OBEX)”, Bluetooth SIG, available at: <http://www.bluetooth.com/English/Technology/Works/pages/obex.aspx>, last visited Mai 2010.

[60]. K. Scarfone, J. Padgette, “Guide to Bluetooth Security”, National Institute of Standards and Technology (NIST), NIST Special Publication 800-121, September 2008, available at: <http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf>

[61]. J. M. Johansson, “Windows Server 2008 Security Resource Kit”, Microsoft Press, 2008

[62]. M. E. Russinovich, D. A. Solomon, “Microsoft Windows Internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000”, 4th ed., Microsoft Press, 2005

[63]. D. Todorov, ”Mechanics of user identification and authentication: Fundamentals of Identity Management”, Auerbach Publications, 2007

[64]. “How Interactive Logon Works”, Windows TechNet Library, 2009, available at: <http://technet.microsoft.com/en-us/library/cc780332%28WS.10%29.aspx#w2k3tr_intlg_how_tpxs>

[65]. “Access Control Model”, MSDN Library, 2009, available at: <http://msdn.microsoft.com/en-us/library/aa374876%28VS.85%29.aspx>

[66]. “Access Mask Format”, MSDN Library, 2009, available at: <http://msdn.microsoft.com/en-us/library/aa374896%28VS.85%29.aspx>

[67]. “Access Control Lists”, MSDN Library, 2009, available at: <http://msdn.microsoft.com/en-us/library/aa374872%28VS.85%29.aspx>

[68]. “How DACLs Control Access to an Object”, MSDN Library, 2009, available at: <http://msdn.microsoft.com/en-us/library/aa446683%28VS.85%29.aspx>

[69]. “Privileges”, MSDN Library, 2009, available at: <http://msdn.microsoft.com/en-us/library/aa379306%28VS.85%29.aspx>

[70]. R. Morimoto, M. Noel, O. Droubi, R. Mistry, C. Amaris, “Windows Server 2008 Unleashed”, Sams, 2008

[71]. “Application Compatibility: Session 0 Isolation”, MSDN Library, 2009, available at : <http://msdn.microsoft.com/en-us/library/bb756986.aspx>

[72]. M. E. Russinovich, D. A. Solomon, A. Ionescu, “Windows Internals: Including Windows Server 2008 and Windows Vista”, 5th ed., Microsoft Press, 2009

[73]. “Windows Interactive Logon Architecture” ", Microsoft Corporation, 2010, available at: <http://technet.microsoft.com/en-us/library/ff404303(WS.10).aspx>

[74]. "Windows Vista Smart Card Infrastructure", Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/bb905527.aspx>

[75]. "SSP Packages Provided by Microsoft", Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/aa380502(VS.85).aspx>

[76]. "Windows Authentication", Microsoft Corporation, 2010, available at: <http://technet.microsoft.com/en-us/library/cc755284(WS.10).aspx>

[77]. "Microsoft Negotiate", Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/aa378748(VS.85).aspx>

[78]. "Logon and Authentication Technologies", Microsoft Corporation, 2003, available at: <http://technet.microsoft.com/en-us/library/cc780455(WS.10).aspx>

[79]. "Introducing Extensions to the Negotiate Authentication Package", Microsoft Corporation, 2009, available at: <http://technet.microsoft.com/en-us/library/dd560645(WS.10).aspx>

[80]. ”Windows Vista Credential Providers", Microsoft Corporation, Credential Provider Technical Reference, 2006, available at:< http://go.microsoft.com/fwlink/?LinkId=93340>

[81]. “Understanding Logon and Authentication”, 2005, available at: <http://technet.microsoft.com/en-us/library/bb457114.aspx>

[82]. L. Zhu, B. Tung, “Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)”, RFC 4556, 2006, available at: <http://tools.ietf.org/html/rfc4556>

[83]. A. Nirmalananthan, "The Smart Card Cryptographic Service Provider Cookbook ", Microsoft Corporation, 2002, available at: <http://msdn.microsoft.com/en-us/library/ms953432.aspx>

[84]. “What's New in Smart Cards”, 2009, available at: <http://technet.microsoft.com/en-us/library/dd367851%28WS.10%29.aspx>

[85]. D. Griffin, “Create Custom Login Experiences With Credential Providers For Windows Vista”, MSDN magazine, 2007, available at: <http://msdn.microsoft.com/en-us/magazine/cc163489.aspx>

[86]. “Windows Vista Sample Credential Providers Overview”, Microsoft Corporation, 2006, available at:< http://www.microsoft.com/downloads/details.aspx?FamilyID=b1b3cbd1-2d3a-4fac-982f-289f4f4b9300&displaylang=xh >

[87]. "Custom Security Packages", Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/aa375200(VS.85).aspx>

[88]. "Creating Custom Authentication Packages", Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/aa375200(VS.85).aspx>

[89]. “Smart Card Subsystem Architecture”, Windows TechNet Library, 2010, available at: <http://technet.microsoft.com/en-us/library/ff404300(WS.10).aspx>

[90]. “CNG Key Storage Providers”, Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/bb931355(VS.85).aspx>

[91]. “CNG features”, Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/bb204775(v=VS.85).aspx>

[92]. “Certificates overview”, Microsoft Corporation, MSDN Library, 2010, available at: <http://msdn.microsoft.com/en-us/library/ee498505.aspx>

[93]. “Base Provider Algorithms”, Microsoft Corporation, MSDN Library, 2010, available at: <http://msdn.microsoft.com/en-us/library/aa375599(v=VS.85).aspx>

[94]. “Understanding Cryptographic Providers”, Microsoft Corporation, MSDN Library, 2009, available at:< http://msdn.microsoft.com/en-us/library/bb931380(v=VS.85).aspx>

[95]. “About Personal Identity Verification (PIV) of Federal Employees and Contractors”, National Institute of Standards and Technology, 2008, available at: <http://csrc.nist.gov/groups/SNS/piv/index.html>

[96]. “Smart Card Mini-driver Specification”, v.7.06, Microsoft Corporation , 2009, available at: <http://download.microsoft.com/download/7/E/7/7E7662CF-CBEA-470B-A97E-CE7CE0D98DC2/sc-mini-driver_specs_V7.docx>

[97]. “Certificate Enumeration”, Windows TechNet Library, 2010, available at: <http://technet.microsoft.com/en-us/library/ff404289(WS.10).aspx>

[98]. K. Ritvanen, K. Nyberg, “Key Replay Attack on Improved Bluetooth Encryption”, Nokia Research Center, available at:< http://www.tcs.hut.fi/Publications/knyberg/kaarle.pdf>

[99]. I. Jorstad, Do Van Thanh, “The Mobile Phone as Authentication Token”,

[100]. “Offering SIM Strong Authentication to Internet Services”, available at: <http://www.ongx.org/SIM_STRONG_WHITEPAPER2.3_SMS.Screen.pdf>

[101]. “Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)”, RFC 4186, available at: <http://tools.ietf.org/html/rfc4186>

[102]. “Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)”, RFC 4187, available at: <http://tools.ietf.org/html/rfc4187>

[103]. S. Hallsteinsen, I. Jorstad, Do Van Thanh, “Using the mobile phone as a security token for unified authentication”, International Conference on Signal Processing,

Communication and Networking (ICSCN), August 2007

[104]. “HOTP: An HMAC-Based One-Time Password Algorithm”, RFC 4226, available at: <http://tools.ietf.org/html/rfc4226 >

[105]. “TOTP: Time-based One-time Password Algorithm”, Internet-Draft, available at: < http://tools.ietf.org/html/draft-mraihi-totp-timebased-03 >

[106]. “OCRA: OATH Challenge-Response Algorithms”, Internet-Draft, available at: < http://tools.ietf.org/html/draft-mraihi-mutual-oath-hotp-variants-09 >

[107]. “Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)”, RFC 4556, 2006, <http://tools.ietf.org/html/rfc4556>

[108]. “Smart Card Driver Library”, Microsoft Corporation, MSDN Library, available at: <http://msdn.microsoft.com/en-us/library/ff548990(VS.85).aspx>

[109]. “OTP Pre-authentication”, IETF Internet-Draft, October 2009, available at: < http://tools.ietf.org/html/draft-ietf-krb-wg-otp-preauth-11>

[110]. “A Generalized Framework for Kerberos Pre-Authentication”, IETF Internet-Draft, October 2009, available at: <http://tools.ietf.org/html/draft-ietf-krb-wg-preauth-framework-15>

[111]. “Mobile One Time Passwords”, v.1.07, available at:< http://motp.sourceforge.net/>, last visited May 2010

[112]. ”ActivIdentity Introduces PKI Secure Mobile Solution for Smart Phones”, ActivIdentity Corporation, available at: <http://www.actividentity.com/company/news/details/?RID=598>, last visited May 2010

[113]. “EZMCOM launches MSIGN: Mobile PKI Platform”, May 2010, available at: <http://www.pressreleasepoint.com/ezmcom-launches-msign-mobile-pki-platform>

7. **Appendix A:** source code for the Bluetooth smart card reader architecture

This section contains UML 2.1 diagrams for the L2CAPClient MIDlet and the source code for JCpki applet from the Bluetooth smart card reader architecture. The source code of the driver itself and of the L2CAPClient MIDlet is on the enclosed CD.

7.1 L2CAPClient UML diagrams

The following picture depicts the Java L2CAPClient MIDlet class diagram.

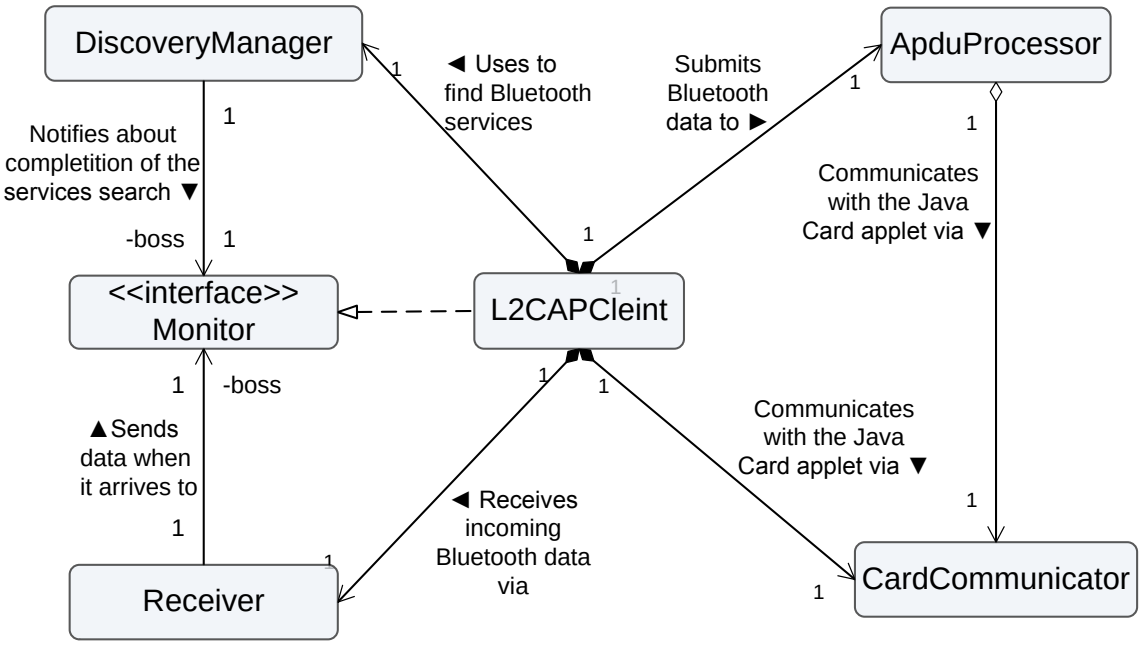

Figure 7.1.1: L2CAPClient class diagram

The detailed description of these classes is provided below.

DiscoveryListener

DiscoveryManager

- boss: Monitor
- discoveryAgent: DiscoveryAgent
- localDevice: LocalDevice
- servicesList: UUID ([])
- serviceRecord: ServiceRecord ([])
- remoteBthDevices: Vector

+ DiscoveryManager (Monitor)
+ performServiceSearch (UUID[]) : void
+ deviceDiscovered (RemoteDevice, DeviceClass) : void
+ inquiryCompleted (int) : void
- searchServices (UUID[]) : void
+ servicesDiscovered (int, ServiceRecord[]) : void
+ serviceSearchCompleted (int, int) : void

CardCommunicator

- cardConnection: APDUConnection

~ SW_NO_ERROR: byte ([]) {readOnly}
~ INS_VERIFY_PIN: byte {readOnly}
~ INS_CHANGE_PIN: byte {readOnly}

+ CardCommunicator(APDUConnection)
+ sign(byte[]) : byte[]
+ updateKeys() : boolean
+ verifyPIN(String) : boolean
+ updatePIN(String) : boolean
+ updatePIN(byte[]) : boolean

- makePinApdu(byte, String) : byte[]
- makePinApdu(byte, byte[]) : byte[]
- makeSignatureApdu(byte[]) : byte[]

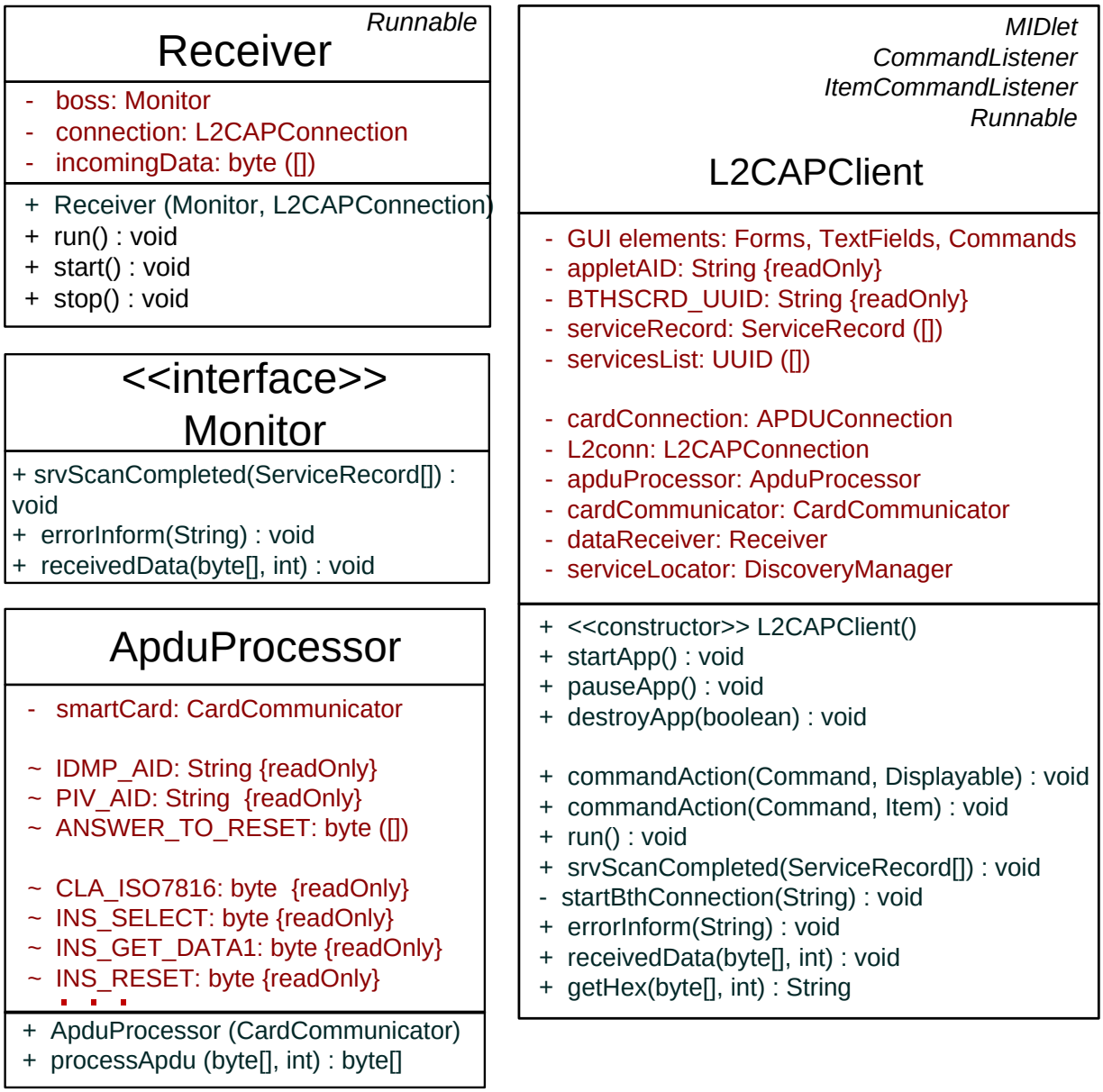

Figure 7.1.2: L2CAPClient classes

7.2 JCpki Java Card applet's source code

The following section contains the JCpki Java Card applet's source code.

```
1  /*
2   * Copyright (c) 2010 NTNU/KTH author: Oleksandr Bodriagov
3   
4   * JCpki.java    15/04/2010
5   * This applet manages RSA public-private key pair and signs data according to sha-
      1WithRSAEncryption algorithm.
6   * !Cryptographic operations may not be fully supported in some simulators
7   */
8  package pki;
9  
10 import javacard.framework.APDU;
11 import javacard.framework.ISO7816;
```

```
12 import javacard.framework.ISOException;
13 import javacard.framework.Applet;
14 import javacard.framework.OwnerPIN;
15 import javacard.framework.JCSystem;
16 import javacard.framework.Util;
17 import javacard.security.*;
18
19 public class JCpki extends Applet
20 {
21    final static byte VERIFY_PIN = (byte) 0x20; // INS value for ISO 7816-4 VERIFY
22    final static byte UPDATE_PIN = (byte) 0x57; // INS value for update PIN request
23    final static byte SIGN = (byte) 0x59; // INS value for the Signature operation
24    final static byte GET_PUBLIC_EXPONENT = (byte) 0x54; // INS value
25    final static byte GET_PUBLIC_MODULUS = (byte) 0x55; // INS value
26    final static byte UPDATE_KEYS = (byte) 0x46; // INS ISO GENERATE ASYMMETRIC KEY PAIR
27    final static byte [] SW_NO_ERROR = {(byte)0x90,(byte)0x00};
28
29    final static byte TryLimit = (byte) 4;// number of tries for PIN
30    final static byte MaxPINSize = (byte) 8;
31    private KeyPair rsaPair;
32    private OwnerPIN appletPIN;
33    private RSAPrivateKey rsaPrivateKey;
34    private RSAPublicKey rsaPublicKey;
35    private Signature mSignature;
36
37    private short dataLength;
38    private byte[] signedbuffer; //array for storing signed data
39    private short signatureLength;
40
41    // Constructor. Only this class's install method should create the applet object.
42    protected JCpki(byte[] installParameters, short offset, byte length)
43    {
44       byte [] initialPIN = new byte[4];
45       short pinOffset = offset;
46
47       Util.arrayCopy(installParameters, (short)(pinOffset), initialPIN,(short)0 , (short)4);
48       appletPIN = new OwnerPIN(TryLimit, MaxPINSize);
49       appletPIN.update(initialPIN, (short) 0,(byte) 4);
50
51       //generate public-private key pair; simulator supports only 512-bit key
52       //rsaPair = new KeyPair( KeyPair.ALG_RSA, KeyBuilder.LENGTH_RSA_2048 );
53       rsaPair = new KeyPair( KeyPair.ALG_RSA, (short)512 );
54       rsaPair.genKeyPair();
55       rsaPublicKey = (RSAPublicKey) rsaPair.getPublic();
56       rsaPrivateKey = (RSAPrivateKey) rsaPair.getPrivate();
57
58       //initialize signature
59       mSignature = Signature.getInstance(Signature.ALG_RSA_SHA_PKCS1, false);
60       mSignature.init(rsaPrivateKey, Signature.MODE_SIGN);
61
62       //the buffer is cleared when applet is deselected
63       signedbuffer = JCSystem.makeTransientByteArray((short)128,
                                                       JCSystem.MEMORY_TYPE_TRANSIENT_RESET);
64       register();
65    }
66
67    // Installs applet. installParameters: contains installation parameters ( PIN, )
68    public static void install(byte[] bArray, short bOffset, byte bLength)
69    {
70       new JCpki(bArray, bOffset, bLength);
71    }
72
73
74
75    // Processes incoming APDUs
76    public void process(APDU apdu)
```

```
77    {
78       byte[] buffer = apdu.getBuffer(); //gets a pointer to the APDU buffer
79
80       // check if it is SELECT APDU command
81       if (buffer[ISO7816.OFFSET_CLA] == 0 && buffer[ISO7816.OFFSET_INS] == (byte) (0xA4))
82       return;
83
84       switch (buffer[ISO7816.OFFSET_INS])
85       {
86           case VERIFY_PIN:
87                 verifyPIN(apdu);
88                 break;
89           case SIGN:
90                 signData(apdu);
91                 break;
92           case UPDATE_PIN:
93                 updatePIN(apdu);
94                 break;
95           case GET_PUBLIC_EXPONENT:
96                 sendExponent(apdu);
97                 break;
98           case GET_PUBLIC_MODULUS:
99                 sendModulus(apdu);
100                    break;
101           case UPDATE_KEYS:
102                 updateKeys(apdu);
103                 break;
104           default:
105                 ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
106       }
107    }
108
109    // user authentication via PIN check
110    private void verifyPIN(APDU apdu)
111    {
112       byte[] buffer = apdu.getBuffer();
113       byte dataLength = buffer[ISO7816.OFFSET_LC];//number of bytes in the APDU's data field
114       if (dataLength > 8 || dataLength < 1)
115          ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
116       apdu.setIncomingAndReceive(); //data is read into apdu buffer
117
118       if (appletPIN.check(buffer, ISO7816.OFFSET_CDATA, dataLength ) != true)
119             ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
120       else
121       {
122          apdu.setOutgoing();
123          apdu.setOutgoingLength((short) SW_NO_ERROR.length);
124          apdu.sendBytesLong(SW_NO_ERROR, (short) 0, (short)SW_NO_ERROR.length);
125       }
126    }
127
128    private void updatePIN(APDU apdu)
129    {
130       if(appletPIN.isValidated() != true)
131          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
132       appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
133
134       byte[] buffer = apdu.getBuffer();
135       byte dataLength = buffer[ISO7816.OFFSET_LC];//number of bytes in the APDU's data field
136       apdu.setIncomingAndReceive(); //data is read into the apdu buffer
137
138       if (dataLength > 8 || dataLength < 4)
139          ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
140       appletPIN.update(buffer, ISO7816.OFFSET_CDATA, dataLength);
141       appletPIN.resetAndUnblock();
142
```

```
143       apdu.setOutgoing();
144       apdu.setOutgoingLength((short) SW_NO_ERROR.length);
145       apdu.sendBytesLong(SW_NO_ERROR, (short) 0, (short)SW_NO_ERROR.length);
146       }
147
148
149    private void sendExponent(APDU apdu)
150    {
151       if(appletPIN.isValidated() != true)
152          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
153       appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
154       try
155       {
156          dataLength = rsaPublicKey.getExponent(signedbuffer, (short) 0);
157       }
158       catch (CryptoException e)
159       {
160          ISOException.throwIt(e.getReason());
161          return;
162       }
163       apdu.setOutgoing();
164       apdu.setOutgoingLength((short) dataLength);
165       apdu.sendBytesLong(signedbuffer, (short) 0, (short) dataLength);
166    }
167
168
169    private void sendModulus (APDU apdu)
170    {
171       if(appletPIN.isValidated() != true)
172          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
173       appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
174       try
175       {
176          dataLength = rsaPublicKey.getModulus(signedbuffer, (short)0);
177       }
178       catch (CryptoException e)
179       {
180          ISOException.throwIt(e.getReason());
181          return;
182       }
183       apdu.setOutgoing();
184       apdu.setOutgoingLength((short) dataLength);
185       apdu.sendBytesLong(signedbuffer, (short) 0, (short) dataLength);
186    }
187
188
189    private void updateKeys(APDU apdu)
190    {
191       if(appletPIN.isValidated() != true)
192          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
193       appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
194       try
195       {
196          rsaPair.genKeyPair();
197          rsaPublicKey = (RSAPublicKey) rsaPair.getPublic();
198          rsaPrivateKey = (RSAPrivateKey) rsaPair.getPrivate();
199       }
200       catch (CryptoException e)
201       {
202          ISOException.throwIt(e.getReason());
203          return;
204       }
205       apdu.setOutgoing();
206       apdu.setOutgoingLength((short) SW_NO_ERROR.length);
207       apdu.sendBytesLong(SW_NO_ERROR, (short) 0, (short)SW_NO_ERROR.length);
208    }
```

```
209
210
211    private void signData(APDU apdu)
212    {
213       if(appletPIN.isValidated() != true)
214          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
215       appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
216
217       byte[] buffer = apdu.getBuffer();
218       dataLength = buffer[ISO7816.OFFSET_LC]; // number of bytes in the data field of APDU
219       apdu.setIncomingAndReceive(); //data is read into the Apdu buffer
220       try
221       {
222             signatureLength = mSignature.sign(buffer, (short)ISO7816.OFFSET_CDATA,
                                                    dataLength, signedbuffer, (short) 0);
223       }
224       catch (CryptoException e)
225       {
226             ISOException.throwIt(e.getReason());
227             return;
228       }
229       apdu.setOutgoing();
230       apdu.setOutgoingLength((short) signatureLength);
231       apdu.sendBytesLong(signedbuffer, (short) 0, signatureLength);
232    }
233 }//end class JCpki
```

8. Appendix B: source code for the TOTP scheme

This section contains UML 2.1 diagrams for the TOTPClient MIDlet and the source code for JCtotp applet. The source code of the MIDlet itself is on the enclosed CD.

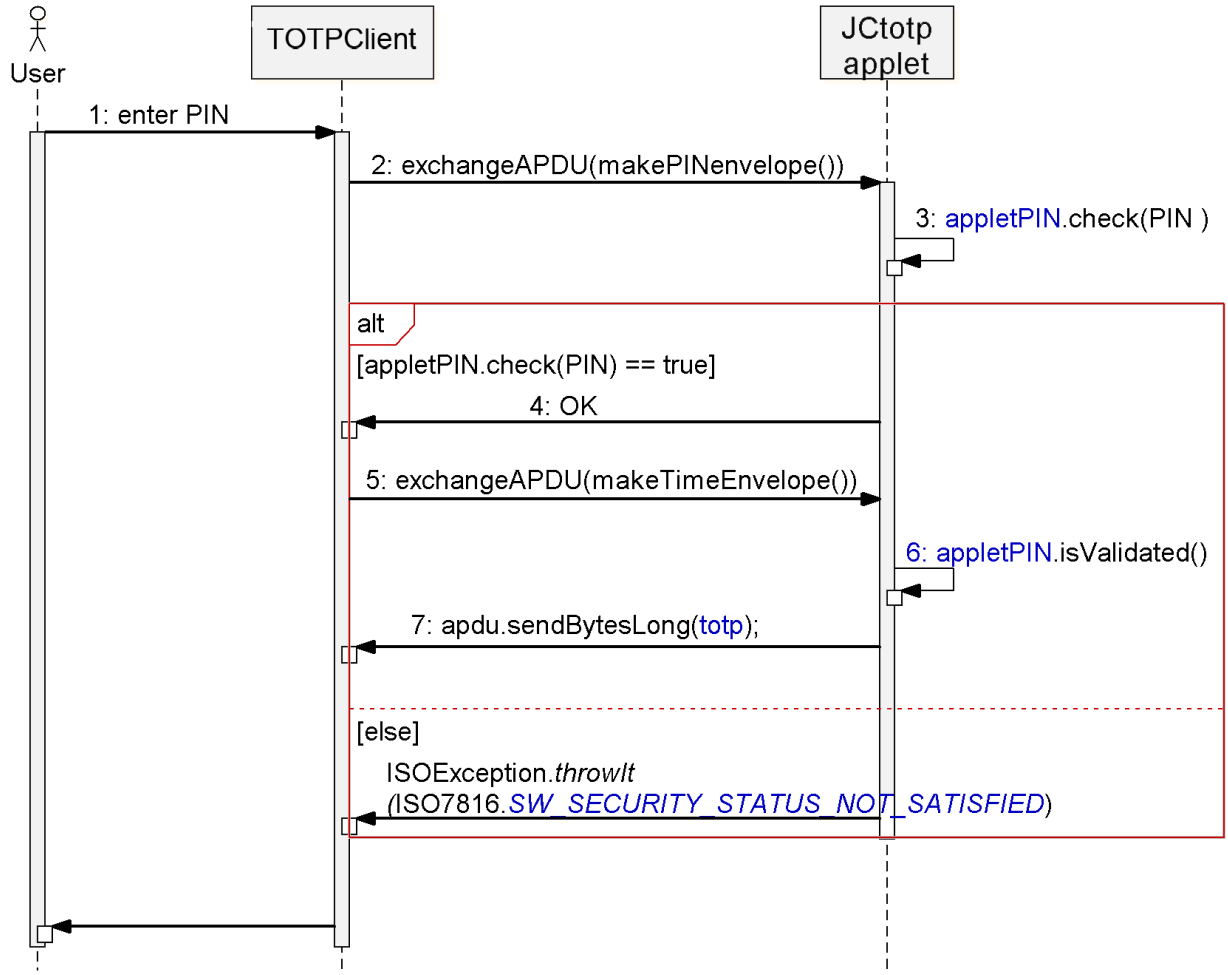

Figure 8.1: TOTP sequence diagram

8.1 TOTPClient Class diagram

The following diagram depicts the Java TOTPClient class diagram.

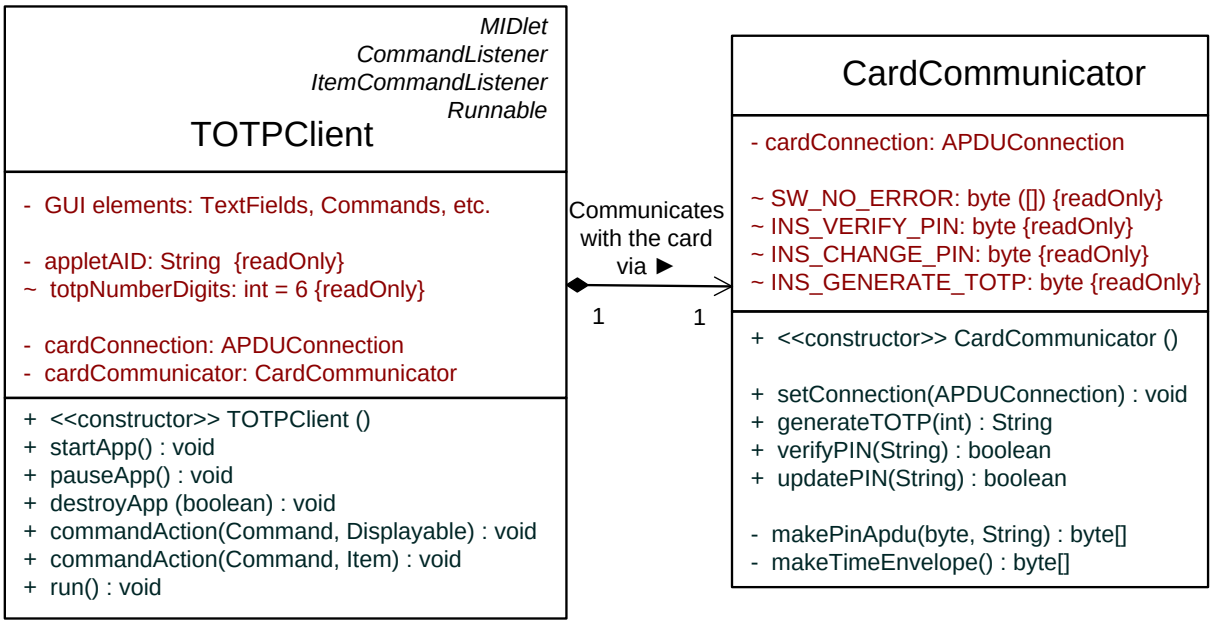

Figure 8.1.1: TOTPClient class diagram

8.2 JCtotp applet's source code

The following section contains the JCtotp Java Card applet's source code.

```
/*
 * Copyright (c) 2010 NTNU, KTH author Oleksandr Bodriagov.

 *  JCtotp.java  05/03/2010

 *  This class provides secure HMAC-based One Time Password generation
 *  based on the shared secret K and the data received from the phone (time value).
 *  Returned value is a 32-bit byte array.
 *  The secret key and the initial PIN are transferred to the applet during applet
 *  loading/installation
 */

package totp;

import javacard.framework.APDU;
import javacard.framework.ISO7816;
import javacard.framework.ISOException;
import javacard.framework.Applet;
import javacard.framework.OwnerPIN;
import javacard.framework.JCSystem;
import javacard.framework.Util;
import javacard.security.MessageDigest;

public class JCtotp extends Applet
{
    final static byte VERIFY_PIN = (byte) 0x20; // INS value for ISO 7816-4 VERIFY
    final static byte UPDATE_PIN = (byte) 0x57; // INS value for update PIN request
    final static byte GENERATE_TOTP = (byte) 0x58; // INS value in the generate-request APDUs
    final static byte [] SW_NO_ERROR = {(byte)0x90,(byte)0x00};

    final static byte TryLimit = (byte) 4;
```

```
32    final static byte MaxPINSize = (byte) 8;
33    final static byte secKeyLength = (byte) 32;//in bytes
34    private OwnerPIN appletPIN;
35    private byte [] secretKey;
36
37    final static byte blockLength = 64;//block lengths in HMAC-SHA1
38    private byte [] workbuffer; // buffer for HMAC-SHA method
39    private short offset;// offset value from which HOTP truncation is started
40    private byte[] hotp; //array for storing the final HOTP value
41    private MessageDigest shaDigest;
42
43    // Constructor. initializes secret key & PIN variables, and data buffers
44    protected JCtotp(byte[] installParameters, short offset, byte length)
45    {
46       byte [] initialPIN = new byte[4];
47       short pinOffset = offset;
48       secretKey = new byte[secKeyLength];
49
50       Util.arrayCopy(installParameters, (short)(pinOffset), initialPIN,(short)0 , (short)4);
51       Util.arrayCopy(installParameters, (short)(pinOffset + 4), secretKey, (short)0,
                        secKeyLength);
52
53       appletPIN = new OwnerPIN(TryLimit, MaxPINSize);
54       appletPIN.update(initialPIN, (short) 0,(byte) 4);
55
56       shaDigest = MessageDigest.getInstance(MessageDigest.ALG_SHA,false);
57       hotp = JCSystem.makeTransientByteArray((short)20,
                                                JCSystem.MEMORY_TYPE_TRANSIENT_RESET);
58       workbuffer = JCSystem.makeTransientByteArray((short)(blockLength +
                                          MessageDigest.LENGTH_SHA),
59                                        JCSystem.MEMORY_TYPE_TRANSIENT_RESET);
60
61       register();
62    }
63
64    // Installs applet. installParameters: contains installation parameters (PIN ,secret key, )
65    public static void install(byte[] bArray, short bOffset, byte bLength)
66    {
67       new JCtotp(bArray, bOffset, bLength);
68    }
69
70    // Processes incoming APDUs
71    public void process(APDU apdu)
72    {
73       byte[] buffer = apdu.getBuffer(); //gets a pointer to the APDU buffer
74
75          // check if it is SELECT APDU command
76       if (buffer[ISO7816.OFFSET_CLA] == 0 && buffer[ISO7816.OFFSET_INS] == (byte)(0xA4))
77          return;
78
79       switch (buffer[ISO7816.OFFSET_INS])
80       {
81           case VERIFY_PIN:
82                verifyPIN(apdu);
83                break;
84           case GENERATE_TOTP:
85                generateTOTP(apdu);
86                break;
87           case UPDATE_PIN:
88                updatePIN(apdu);
89                break;
90           default:
91                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
92       }
93    }
94
```

```
95    // user authentication via PIN check
96    private void verifyPIN(APDU apdu)
97    {
98       byte[] buffer = apdu.getBuffer();
99       byte dataLength = buffer[ISO7816.OFFSET_LC];// number of bytes in the APDU's data field
100      if (dataLength > 8 || dataLength < 1)
101         ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
102      apdu.setIncomingAndReceive(); //data is read into apdu buffer
103
104      if (appletPIN.check(buffer, ISO7816.OFFSET_CDATA, dataLength ) != true)
105            ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
106      else
107      {
108         apdu.setOutgoing();
109         apdu.setOutgoingLength((short) SW_NO_ERROR.length);
110         apdu.sendBytesLong(SW_NO_ERROR, (short) 0, (short)SW_NO_ERROR.length);
111      }
112   }
113
114   private void updatePIN(APDU apdu)
115   {
116      if(appletPIN.isValidated() != true)
117         ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
118      appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
119
120      byte[] buffer = apdu.getBuffer();
121      byte dataLength = buffer[ISO7816.OFFSET_LC];//number of bytes in the APDU's data field
122      apdu.setIncomingAndReceive(); //data is read into apdu buffer
123
124      if (dataLength > 8 || dataLength < 4)
125         ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
126      appletPIN.update(buffer, ISO7816.OFFSET_CDATA, dataLength);
127      appletPIN.resetAndUnblock();
128
129      apdu.setOutgoing();
130      apdu.setOutgoingLength((short) SW_NO_ERROR.length);
131      apdu.sendBytesLong(SW_NO_ERROR, (short) 0, (short)SW_NO_ERROR.length);
132   }
133
134   private void generateTOTP(APDU apdu)
135   {
136      if(appletPIN.isValidated() != true)
137         ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
138      appletPIN.reset();//resets the validated flag and resets the PIN try counter to the value
139
140      byte[] buffer = apdu.getBuffer();
141      byte dataLength = buffer[ISO7816.OFFSET_LC];
142      if (dataLength > 8)
143         ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
144      apdu.setIncomingAndReceive(); //data is read into apdu buffer
145
146    limitedHmacSha(hotp, secretKey, buffer, short)ISO7816.OFFSET_CDATA,(short)dataLength);
147
148      // Dynamic truncation according to RFC 4226
149      offset = (short)(hotp[MessageDigest.LENGTH_SHA - 1] & 0xf); //take only 4 low-order bits
150
151      apdu.setOutgoing();
152      apdu.setOutgoingLength((short) 4);
153      apdu.sendBytesLong(hotp, (short) offset, (short) 4);
154   }
155
156   //calculates HMAC-SHA1 according to RFC 2104 for data blocks <= 20 bytes
157   private void limitedHmacSha (byte [] mac, byte [] key,  byte [] data, short dOffset,
                                   short datalength)
158   {
159      byte i = 0;
```

```
160       for (i = 0; i < blockLength; i++)
161         workbuffer[i] = (byte)0x00;
162
163       //hash the key if it is longer than 64 bytes and write result to workbuffer
164       if (key.length > blockLength)
165         shaDigest.doFinal(key, (short) 0, (short)key.length, workbuffer, (short) 0);
166       else
167         for (i = 0; i < key.length; i++)
168         workbuffer[i] = key[i];
169
170       //K XOR ipad
171       for (i = 0; i < blockLength; i++)
172          workbuffer [i] ^= (byte) 0x36;
173
174       //append the stream of data 'text' to the (K XOR ipad)
175       for (i = 0; i < datalength; i++)
176          workbuffer[(short)(blockLength + i)] = data [(short)(dOffset + i)];
177
178        //creates H(K XOR ipad, text) and appends to the buffer after the space for (K XOR opad)
179       shaDigest.doFinal(workbuffer, (short) 0, (short)(blockLength + datalength), workbuffer,
                          (short) blockLength);
180
181       //calculates (K XOR opad) and puts it in the buffer before H(K XOR ipad, text)
182       for (i = 0; i < blockLength; i++)
183       {
184          workbuffer [i] ^= (byte) 0x36;//take away previous XOR
185          workbuffer [i] ^= 0x5C;
186       }
187       // calculates H(K XOR opad, H(K XOR ipad, text)) and writes it to mac
188       shaDigest.doFinal (workbuffer, (short) 0, (short) (blockLength +
                             MessageDigest.LENGTH_SHA), mac, (short) 0);
189    }
190 }//end class JCtotp
```

9. Appendix C: digital attachments

The attached ZIP archive contains the following directories and files:

- Bluetooth smart card reader architecture:
  - Bluetooth smart card reader driver
    - driver built for Windows7_x86
    - source
    - driver installation instructions.pdf
    - driver installation instructions.txt
  - L2CAPClient
    - dist
    - source
  - JCpki
    - source
    - jcpki.script
    - simulation instructions for JCDK 3.02.txt
- TOTP architecture:
  - TOTPClient
    - dist
    - source
  - JCtotp
    - source
    - jctotp.script
    - simulation instructions for JCDK 3.02.txt
- Master's thesis in pdf format