UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Implementation Trade-offs for Access Tokens

Dissertation presented by
Nicolas COGNAUX

for obtaining the Master's degree in
Electrical Engineering

Supervisor(s)
François-Xavier STANDAERT , François KOEUNE , François MACÉ

Reader(s)
Edouard CUVELIER

Academic year 2015-2016

This thesis is the conclusion of one year of work. This year gave me opportunities to better understand security of access authentication systems in practice. I already had theoretical basis given by classes I attended during my Master. Being able to dig into the practical aspects is a real advantage. I hence have to thank NCS Scaline and more particularly François Macé, who provided me this thesis subject and the experimental setup. I also accessed confidential datasheets that allowed me to better understand implementations and realities of the industrial world.

My academic promoters François-Xavier Standaert and François Koeune also played a great role in the achievement of this thesis: They gave me a direction to follow and guided me to the right questions in order to touch interesting subjects. They also helped me with theoretical aspects of Pairing Based Cryptography and Elliptic Curves Cryptography. Those are more mathematical subjects that I almost discovered with this thesis. I sometimes needed a more Mathematical background that they were able to provide in order to understand keys of those abstracted subjects. They also gave me interesting documents to further understand those topics and improve my thesis.

Different people read and corrected this document, they simplified sentences and explanations that where not easily comprehensive. I took the decision to write this thesis in English which is a great exercise but also sometimes a bit complicated. I hope those corrections given allowed me to improve my written English. I thus also thank those readers.

Finally, I have to thank the University which gave me those five years of intense learnings during which I discovered a real passion for Electronics and particularly Information Security. Those previously obscure subjects are now my future career topics and I hope to continue learning in those directions.

Acknowledgements

Confidential documents

This thesis concerns protected materials from NXP and STid. Confidential documents have been used for this work, those are listed in the bibliography and are subject to NDAs.

Technical explanations present in this thesis are related to public documents or open standards and nothing was published from those confidential documents. Those only served as reference for the implementation of the different upgrades (Transparent mode).

Licences

Libraries used in Chapter 5 such as GMP and libPBC are open-source and published under the LGPL which is given in appendix **??**.

Trademarks

- Android is a trademark of Google Inc.
- Blackberry is a trademark of BlackBerry Limited.
- Freescale is a trademark of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm.
- iPhone is a trademark of Apple Inc, registered in the US and other countries.
- MIFARE DESFire are registered trademarks of NXP B.V.
- NCS and Scaline are registered trademarks of NCS S.A.
- Qt is a registered trademark of The Qt Company Ltd.
- STid and Architect are registered trademarks of STid SA.
- Windows is a trademark of Microsoft Corporation in the United States and/or other countries.

Contents

Abbreviations

AES Advanced Encryption Standard

AID Application IDentificator

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information

CBC Cipher Block Chaining

CRC Cyclic Redundancy Check

DDH Decisional Diffie-Hellman

DES Data Encryption Standard

ECC Elliptic Curves Cryptography

EEPROM Electrically-Erasable Programmable Read-Only Memory

IV Initiation vector

MAC Message Authentication Code

MVC Model View Controller

NFC Near Field Communication

OS Operating System

OSI Open Systems Interconnection

P2P Peer to Peer

PBC Pairing Based Cryptography

POC Proof of Concept

RA Revocation Authority

RFID Radio Frequency IDentification

RL Revocation List

RSA Rivest-Shamir-Adleman

SDH Strong Diffie Hellman

SDK Software Development Kit

TCP Transmission Control Protocol

UID Unique Identification Digit

VLR Verifier Local Revocation

XDH eXternal Diffie-Hellman

XSGS eXtremely Short Group Signature Scheme

ZKP Zero Knowledge Protocols

List of Figures

List of Tables

Introduction

This thesis focusses on privacy considerations for access authentication systems. Today, access authentication systems are used in every situations and nearly everywhere. Those authentication systems create metadata[1], such as access logs and punctual identifications. By aggregating those metadata which have no value alone, we can guess habits and comportments of users. Current authentication systems can leak those metadata and so compromise privacy. Solutions have to be implemented in order to reduce the amount of those leaks and their relevancy. This thesis covers different systems that can reduce those attacks on privacy.

First chapter of this Thesis exposes current systems and practices in access tokens and access authentication systems. It explains how access tokens work, especially for Radio Frequency IDentification (RFID) tokens. Those are also categorized and compared technically. This chapter also covers some implementation standards for access authentication systems. This thesis also introduces some security risks related to current systems.

Second chapter focus on a particular system that uses MIFARE DESFire EV1 tokens for access authentication. The privacy aspect and protection systems implemented in this token are evoked and discussed. This particular authentication system implements some protections for privacy but those solutions can still be improved. Those weaknesses are discussed and in term of security but also for the privacy of the user. Those systems are often used for cases where privacy is critical although those are not adapted.

A possible improvement, which is the switch to transparent mode for the communication with the readers, is exposed in Third chapter of this document. Transparent mode applied to the MIFARE DESFire EV1 authentication system is an improvement for security and privacy. This chapter covers the improvements but also an implementation of such protocol on an existing authentication system.

However, such systems cannot be used in every situations. For example, some companies use those systems for public transportation whereas the user's privacy is not perfect. Such cases have to use more adapted systems. Group signature is a scheme that can help to improve privacy. This topic is covered in Fourth chapter of this thesis where theoretical information are given. This chapter presents cryptographic schemes that can be used for group signature and authentication. Those schemes are compared theoretically and discussed against practical considerations.

Finally, last chapter presents an implementation of group signature scheme for smartphones. This system is confronted to the use case of a music concerts subscription and is benchmarked for this use case. A proof of concept is also developed and presented in this chapter. It uses Android smartphones as Prover and is aimed to prove the usability and the feasibility of such system with today's technologies.

[1] Data that provides information about other data. Those can be structural metadata and descriptive metadata.

Chapter 1

An introductions to RFID access tokens and authentication systems

1.1 Introduction

Today, access tokens are available in multiple forms: From simple bar-code paper cards to Near Field Communication (NFC) smartphone authentication, we use them every day without even noticing it. This section will briefly summarize the evolution of those tokens and cover their near future. We will also provide some details about their features and the issues they might present. In this section, and in the remainder of this document, we will focus on RFID tokens but also NFC which is a subcategory of RFID.

1.2 RFID Tags background

RFID tags are electronic devices using electromagnetic fieds to communicate. Those are used for identification, authentication and various other applications. In this thesis, we cover the use of RFID tags for authentication, it is hence important to address their specifications.

1.2.1 RFID capabilities

There are multiple types of RFID tokens with multiple capabilities. Figure 1.1 summarizes those flavours and the RFID requirements for different use cases.

As can be seen in Figure 1.1, there are multiple axis that will define an RFID tag itself. Those characteristics depend on the use case. Each characteristic will be defined according to [16].

Power capability

Power can be provided to the tag either by an embedded battery or directly by the reader. In the first case, the tag is said to be **active** as it powers itself via an active electrical source. **Passive** tags only rely on mutual induction and do not use an active power source. The reader sends a powerful magnetic field that is received by the inner coil of the tag and then rectified in order to

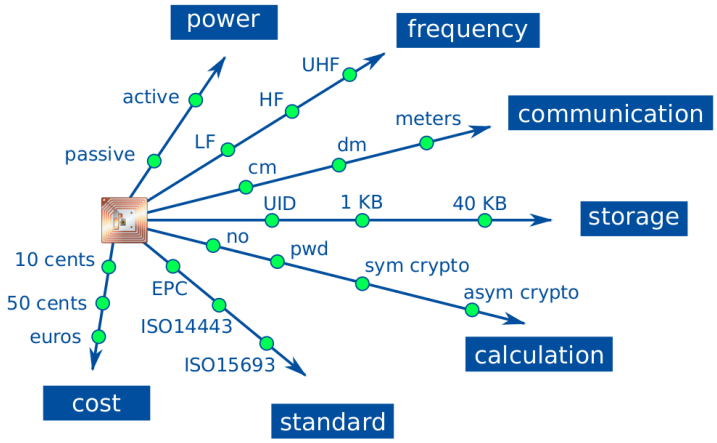

Figure 1.1: Map of RFID characteristics [5]

be used as power supply. This case is often used in small tags that do not need much power as it allows the tag to be extremely low cost.

Hybrid tag versions also exist that use the mutual inductance to power themselves for the computations but rely on an active power supply for the transmission. Those tags are said to be **semi-passive** tags. Those are used for very specific applications and will not be covered in this thesis.

Frequency

The frequency used for the communication between the tag and the reader will also impact the read distance. Higher the frequency, longer is the reach distance. Low frequency tags can be read at a distance of 30 cm where high-frequency tags at the same power can be read from a distance of one meter. But an higher frequency allows using smaller chips and antennas. In a lot of case, it is also possible to reduce the power supply of the transmission for reducing the reach distance in many cases. This last characteristics explains why today's tags are using higher frequencies.

Frequency	Active/Passive	Main use
124 kHz to 135 kHz	Passive or active	Pet identification
13.56 MHz	Passive or active	Smartcards, Payment, Smartphones
860 MHz to 960 MHz	Active	Parking access
2.45 GHz or 5.8 GHz	Active	Toll free and parkings

Table 1.1: Frequencies used by RFID Tags [5]

The different standardized frequencies (from ISO/IEC 18000 [20]) are displayed in table 1.1 with typical use cases and power supply. Passive tags are often built for frequencies comprised between DC (0 Hz) and 30 MHz. At higher frequency, active tags are cheaper and less constrained in term of reach distance. The frequency also plays in the security of the communication: A higher frequency is more costly to listen to. For example, looking at the MIT Access card system, we can observe that one of the vulnerabilities is the low frequency used (125 kHz) [1]. More information are available at page 21 of [16].

It must also be pointed out that higher frequencies are more subject to perturbations from environment. Walls, water, metal, etc. can reduce the reachability of tags relying on higher frequencies [16]. Hence it is important to use them when the reach distance needed is small or in an open environment.

Communication range

The communication range between reader and tag is highly important. In security operations, it is preferable to use smaller ranges (millimetres to centimetres) in order to reduce the spoofing attack[1]. For only identification and tracking, we will prefer wider range as it allows using less antennas and less precision. The range is depending on multiple physical characteristics: the antenna, the transmitting power and the frequency [5].

Computational power and memory

Access tags can have multiple computational capabilities and memory sizes. The smaller ones will include only small amount of read only memory that will simply be broadcasted to readers. The MIT Access card uses this technique and simply broadcasts the Unique Identification Digit (UID) of the card [1]. More evolved tags can compare data sent by the reader and perform simple arithmetic operations. For example passwords or identification tokens can be checked against memorized data. The token simply returns a boolean code if the secret matches[2].

A better computational capability is the ability to perform symmetric cryptography such as Data Encryption Standard (DES), 3DES and Advanced Encryption Standard (AES). In this kind of technology, the tag embeds at least one cryptographic key that will be used to sign cipher or decipher a data generated by the reader. Nowadays, smartcards can embed CPUs for asymmetric cryptography (for schemes such as Diffie Hellman, Rivest-Shamir-Adleman (RSA), etc.). Those cards can also embed bigger memories up to kilobytes of data.

In this document, we will first cover the MIFARE DESFire EV1 (see Chapter 2) card which perform symmetric cryptography (DES, 3DES and AES) and embeds a memory of 2 KB, 4 KB or 8 KB. The second part of the thesis covers NFC enabled Smartphones which are capable of executing complex computations and runs on ARM CPUs.

1.2.2 Standards

In the RFID area, there are a lot of different standards defined for interoperability. The first one, defining how the RFID tags are working and the different frequencies allocated is the ISO/IEC 18000 [20].

NFC is standardized under ISO 14443 [19], this document details the architecture of the system and the communication protocols to use. It is an extension to ISO 18000 that allows more various usages. This standard is also completed by ISO 18092, that allows Peer to Peer (P2P)[3] communications between two initiators. ISO 21481 [22] also completes the NFC standard by adding the ability to emulate a tag. This last capability means that an initiator (e.g. reader, smartphone,...) can act as a passive tag (e.g. a smart card) [2]. This feature will be evoked in section 5.3 as it is widely used when employing smartphones for access authentication via NFC.

[1] This attack consists of listening to communications in order to retro-engineer the system and then forge messages.

[2] The answer from the tag is not really a boolean but a specific code for each of the two possible answers

[3] No initiator/target, the two devices are treated equally.

NFC standards are edited by the NFC forum that is a consortium of different actors that use and implement this technology[4].

1.2.3 Today's challenges around RFID tokens

Privacy

RFID tags are everywhere and more and more difficult to notice for the user. There are multiple privacy issues around them. The big advantage of RFID tags is the reading distance: as tags can be scanned to get information at distances that can go up to 10 meters. But this characteristic is also a potential treat for privacy.

Cheap tags, used for identification of goods in stores and warehouses are identified via a UID[5]. This UID is very often fixed at manufacturing and can be read directly, without any authentication. An attacker can easily scan a public space in order to detect tags, it is then really easy to track them or even to identify them via a database of UID. This database only contains meta-data but those aggregated can lead to real privacy issues.

Forging

In really weak systems, it is also possible to forge a new tag that will be unnoticeable for the reader. We can, for example, cite the MIT Access Card system that was simply broadcasting the UID on a 125 kHz frequency. This card is really simple to forge as the secret data (the UID of the card) is broadcasted publicly at each authentication. Moreover, the frequency used is low and allows to listen easily to the communication. Those two weaknesses allow to forge a new access tag really easily simply by listening one communication. The complete analysis is described in [1]

Relay attacks

Another big threat is the relay attack. It is easy for a smartphone to act as a tag and to simply transmit communication to a legit tag. With the rise of contactless payment, it is pretty easy to relay a tag from a confined public space (e.g. in public transportations) to another device that will act as a legit payment tag. The reader will thus receive legit authentication data as those are generated from a genuine tag. This kind of attack is pretty complicated to detect and defeat and does not cost much [5] [18].

Relay attacks can be really dangerous for electronic payment systems but also for authentication. The only current efficient countermeasure found is to lower the time granted between a message and its answer from the tag. However, the granted time is still too long and cannot really be lowered in order to keep a fully functional system. Further more, standards such as ISO 14443 allows commands to increase the allowed delay for some specific exchanges [19].

Another solution would be to deal with positioning sensors like GPS in order to be sure that the tag is really at the reader vicinity. But in this case, the GPS coordinates have to be authenticated which adds new problems and vulnerabilities. Some works such as [17] covers this solution.

[4]The list can be found at `http://nfc-forum.org/about-us/our-members/`

[5]A 7 byte string that uniquely identify the tag.

1.3 Access control systems

The authentication scheme is almost always the same: Two entities want to be sure that a secret is shared between them. This secret can have multiple forms: a secret key, private data, computed data,...

The Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)[6] published a document summarizing the specifications of an authentication system and the security needs required for different systems [3]. This document is aimed at giving a better understanding of authentication systems for security companies. This section is based on this document in order to cover the background knowledge needed to understand the next chapters of this thesis.

1.3.1 Fundamentals

The aim of an access control system is to filter access to a secured area. This secured area can be a whole facility or a small room. An access control system has to perform three main tasks: Identification and authentication, Data manipulation and Access allowance. Those are detailed below.

Identification and authentication

The identification is the action to give its identity. In a system based on RFID tags, it is often done via the tag UID. The UID is often readable without any authentication but sometimes authentication of the reader is also needed before reading it.

After identification, it is important to authenticate this identity in order to prove it. This authentication can be done at multiple levels: Authentication of the tag or the authentication of the user itself. In the primer case, the tag holds a secret and proves to the Verifier that he knows it. For example, it can sign a specific message using its private key. User authentication can be done using biometric techniques or passwords. Depending on the system and the security level needed, user authentication can be necessary or not. In this thesis, we will only use tag authentication and take the hypothesis that the tag is personal and thus directly linked to the user.

Data manipulation

After authentication, the Verifier is sure of the identity of the tag (and of the user). The decision to allow access is often taken via a database lookup, timer or something else. Further manipulations can be done such as work-time logging. This data manipulation is not in the scope of this Thesis as we particularly focus on authentication.

Allow access

Finally, access is granted, via the opening of a door. It is important to note that this door has to provide several characteristics depending on the specifications. For example, the behaviour in case of power outage has to be defined. This topic is covered by the ANSSI's document [3] but we will not go in further details as this thesis focusses on the enhancement of the authentication process and not the whole system. In our case, we will take the hypothesis that the system is protected against power outages and that access is filtered via a sufficiently secured gantry.

[6] This service is in charge of proposing rules and specifications for the protection of informations and systems adopted in France.

Threat	Who ?	Means	Knowledge	Level
Natural crossing	Curious or unintentional public	No specific means	Nothing	1
Simple logic or mechanical attack	Weak attacker	Publicly available tools	Basic knowledges of the system	2
Evoluted attacks	Prepared attacker	Specific tools	Retro-engineering of the system	3
Sophisticated attack	Well trained attacker	Cryptanalysis and specific tools designed against this system particularly	Retro-engineering and confidential information from the system-provider	4

Table 1.2: Summary of treats and security levels [3]

1.3.2 System designs and architectures

Security requirements

In order to design the system, it is necessary to fix the security requirements in order to define the level of security needed for the system. The security level depends on different parameters and hypothesis:

Secured zones The first parameter is the group of secured zones. The nature of those zones defines the access conditions and the security means to protect them. A secured zone is a specific area of the building or site that has a limited access. For example, in a small company, we can select the offices to be a secured zone and the server room another. Those two areas need different access rights and also have different security requirements because of the value contained in them. The system has to be mainly located in the secured zone.

Potential treats When designing security for a secured zone, the potential treats against the security system have to be defined. Table 1.2 summarizes the treats covered by [3]. When knowing the potential treats and thus the security level needed, it is possible to select the components of the system in order to tackle those treats and to secure de secured zones. Over-protecting a zone is possible but it costs more and can also over constrains the user when trying to access it.

Circulation flows It is also important to fix who can access which zones at what time, via which path,... This flow is used in the Data manipulation step of the access granting. Although it is an important issue to take into account when actually securing facilities, it is not in the scope of our work and will not be discussed further. This Thesis will simply state that authentication is performed before a gate that separates public area and the secured area.

System choices

Based on information given in the ANSSI specifications (see previous subsection), it is now possible to choose the system components and architecture.

Security Level	**Resistance against logical attacks**	**Method**	**Technology**	**Characteristic**
1	None	Tag identification	125kHz tag, ISO 14443 or ISO 15693, no cryptography	Easily clonable and forgeable
2	L1	Tag authentication	ISO 14443 tag with symmetric cryptographic authentication	Authentication based on common master key (eg. 3DES, AES)
3	L2	Tag authentication with diversified keys	ISO 14443 tag with symmetric cryptographic authentication	Authentication based on diversified key (eg. 3DES, AES)
4	L3	Tag authentication and user authentication (eg. password or biometric mean)	ISO 14443 tag with symmetric cryptographic authentication with password or biometric information	Authentication based on diversified key (eg. 3DES, AES)

Table 1.3: Tag technology recommendations [3]

Tags The ANSSI recommends tag technologies based on the security level required. Those recommendations are summarized in table 1.3. This table makes a great correspondence between security level and resistance against logical attacks which defines the security required for every component of the system. The Level of resistance against logical attacks is defined in details for every level in Appendix 4 of Reference [3].

Reader The reader is located at the border between the secured zone and the unsecured area. It is the relay between Prover and Verifier and is thus also subject to security constraints. The reader often contains security elements as we will see in the architecture section. It is important to delete those elements in case of alteration of the device. But in practice, those mechanisms are not always efficient. Also, as the reader is in the unsecured area, we can take the hypothesis that every type of attack is possible such as side channel attacks, spoofing,... Those issues conducted to transparent architectures (Chapter 3 covers a conversion from a classic system to a transparent one) in order to remove the security element from the reader and keep everything in the secured area.

The Access controller (Verifier) The Access controller contains secret informations such as cryptographic elements, user database, access logs,... It has to be located inside the secured area and to be secured physically but also virtually (against logical attacks). The installation and maintenance of the Access controller has to be done carefully as it is the heart of the system. It is also important to connect this device to the minimum viable amount of other devices. Each connection is subject to security flaws: For example, it is highly discouraged to directly connect the Access controller to the Web.

Cryptographic authentication principles

Cryptography is highly used for authentication. The idea of the authentication process is to prove that we have a shared secret. This proof is performed via a Challenge/Response protocol. Meaning that the Verifier (A) sends a challenge to the Prover (B). This challenge is processed by the Prover and the result is sent to the Verifier. The Verifier then checks this result and accepts or denies the authentication. This kind of Challenge/Response protocols are standardized under ISO 9798 and can imply different cryptographic techniques. [21] details Challenge/Response protocols based on cryptography that can be used for authentication using different computational capabilities (symmetric, asymmetric, digital signature,...).

Each part of the document is dedicated to a specific challenge type:

ISO 9798-2 Mechanisms using symmetric encipherment algorithms

ISO 9798-3 Mechanisms using digital signatures techniques

ISO 9798-4 Mechanisms using cryptographic check functions

ISO 9798-5 Mechanisms using zero knowledge techniques

Each part of the standard also details different schemes for the roles of each parties: **Unilateral** or **bilateral** authentication. In the primer case, only the Prover authenticates his identity to the Verifier. In a bilateral authentication, the two parties authenticate with each other. The second authentication brings more security as now both parties are authenticated and are members of the same group (both share the same secret and are thus allowed to communicate). This type of scheme is used when none of the parties can be trusted by the other. Bilateral authentication has to be preferred over unilateral authentication in almost any cases.

System architectures

The ANSSI proposes four different system architectures with specific keys involved. We will cover them briefly as it will be important to explain the first improvement done to the MIFARE DESFire EV1 classic scheme (Evolution between systems of Chapter 2 and Chapter 3).

Two architectures of the four presented here are highly discouraged as those are really insecure. We will begin with them and explain why those are now discouraged.

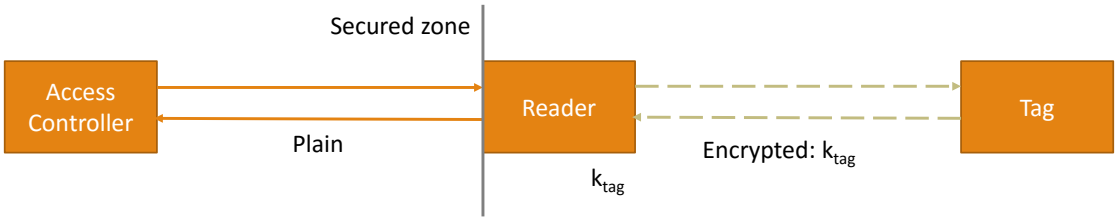

Figure 1.2: Encrypted between Reader and Tag, plain for intra-communication

Encrypted between reader and tag In this architecture, the only encrypted communication is the link between tag and reader (see Figure 1.2). The hypothesis for this architecture is that as the reader is at the border of the secured area, the connection inside the secured area can be considered as secured and so can be a plain communication (not secured). In this case, the tag cannot be cloned simply by listening to exchanges as the communication between the

tag and reader is encrypted. On the other hand, this scheme is highly insecure because secret keys of the tag are stored in the reader (that is considered outside the secured area). Also, the communication is not protected between the Access controller and the reader, meaning that an attacker can simply listen to this channel and get the unprotected communication and thus the secret. Physically, the cabled connection between reader and Access controller can easily be spoofed.

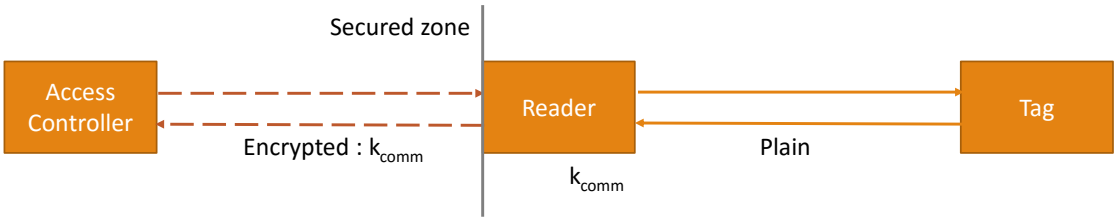

Figure 1.3: Encrypted between Reader and Access controller, plain communication with Tag

Encrypted between reader and Access controller In this architecture, the communication between the Access controller and the reader is secured (see Figure 1.3). This scheme is very unsecured as the secret key to communicate with the Access controller is also stored outside of the secured area (in the reader). Recovering this key (k_{comm}) compromises all decisions based on the communication between the reader and the controller. Sending messages to the channel becomes possible and allows an attacker to fool the controller. Also, the connection between reader and tag is not secured, thus, the security added between reader and Access controller is useless. This also means that communications can be replayed easily.

Encrypted everywhere In this architecture, communications between Access controller and reader are secured as are those between tag and reader (see Figure 1.4). This scheme uses different keys inside the secured area (k_{tag}) and outside (between tag and reader: k_{comm}). This also adds complexity to the system since more keys and cryptographic operations are involved. It is considered secure if the reader is also considered secured. This assumption is crucial because every authentication and cryptographic operations are performed by the reader. It is the default architecture used by many providers and is also the one used for this thesis in Chapter 2. The principal issue with this architecture is the key storage outside the secured area, meaning that those can be recovered via side channel attacks for example.

Transparent mode The transparent mode is the most secured architecture possible that is described in recommendations from ANSSI (see Figure 1.5). The transparent mode uses only

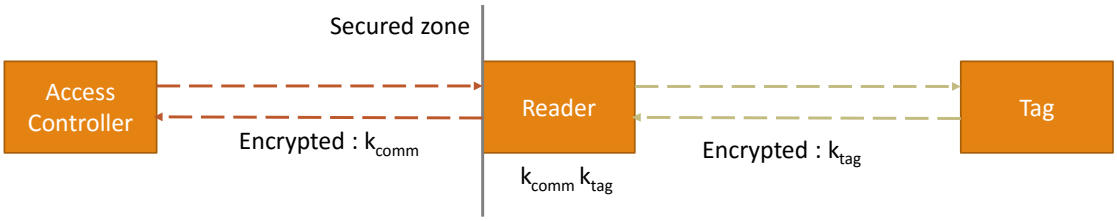

Figure 1.4: Encrypted everywhere

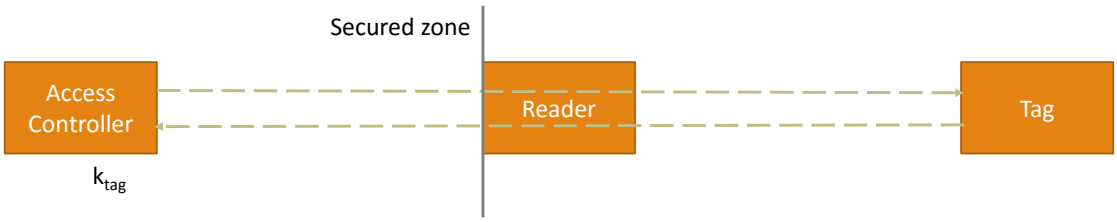

Figure 1.5: Transparent architecture

one key to authenticate the Access controller directly to the tag (k_{tag}). The reader is passive as it only relays data from the tag to the Access controller and vice-versa. The only secret is held by the Access controller (inside the secured area). The Access controller is thus the only possible vulnerability of the system (on a cryptographic point of view). On the other hand, this system implies more computational complexity for the Access controller as it now has to hold the whole authentication protocol. The details of implementation of the transparent architecture is described in Chapter 3 of this Thesis.

1.4 Tags used for this thesis

This section summarizes tags used for the different systems described in this Thesis.

The first type of tags covered in this thesis is passive: we use MIFARE DESFire EV1 cards. Those are used in Sections 2 and 3. In general, active tags are reachable at a greater distance than passive tags. This difference, combined with the cost explain the wide use of passive tags in security and authentication.

Sections about privacy (Sections 4 and 5) relies on active tags via NFC smartphones. The choice of smartphones is guided by the fact that we need more computational capabilities for anonymity.

1.5 Conclusion

This section was a brief introduction to access tokens and RFID tags. It has covered some privacy and security issues faced currently when using RFID tags. It also covered some of the recommendations of the ANSSI for access control systems. We will observe how those recommendations are followed in practice and how to implement some of them (Transparent mode for example) on an actual system.

As we saw, a lot of parameters have to be understood and taken into account when designing an authentication system. We now have the right tools to build a secured system but the system is often badly designed. This bad design conducts to big potential treats.

The use of a MIFARE DESFire EV1 card for authentication and the protections taken against potential privacy and security issues will be covered in the next chapter.

Chapter 2

MIFARE DESFire EV1 access tokens

2.1 Introduction

MIFARE DESFire EV1 cards are one of the latest MIFARE DESFire tokens available from NXP. This RFID token is widely used for Authentication systems but also a wide variety of other applications such as public transportation, ticketing, etc. It is thus interesting to observe systems using this particular token for this thesis. We will first cover the system used for authentication. Then, some weaknesses of this system in particular use cases and possible solutions will be covered.

2.2 System description

This section details the MIFARE DESFire EV1 system and its usage for authentication. Some parts of this section are vague on purpose as the system details are confidential. Documents [32], [34], [31] and [30] are confidential and subject to NDAs, those can be obtained via NXP. Documents [36] and [33] are disclosed publicly on the MIFARE DESFire EV1 and NXP websites. A lot of research also covers retro engineering of MIFARE DESFire cards and are publicly available on Internet.

In this chapter, we will use the case of an authentication system used for access in a small company. We only want an access control and no time-checking capability. Anonymity is not critical inside the company (between employees and the Access controller).

2.2.1 System architecture

The system is composed of four entities. Those are standards entities for authentication systems and were explained in the previous chapter:

Verifier The Verifier checks the identity of the Prover. In almost every case, it is located in the secured area and is composed of an embedded system or a more powerful computer depending on the computational needs. In our case, the embedded system used is the SC415 from NCS Scaline see Appendix **??** for technical details.

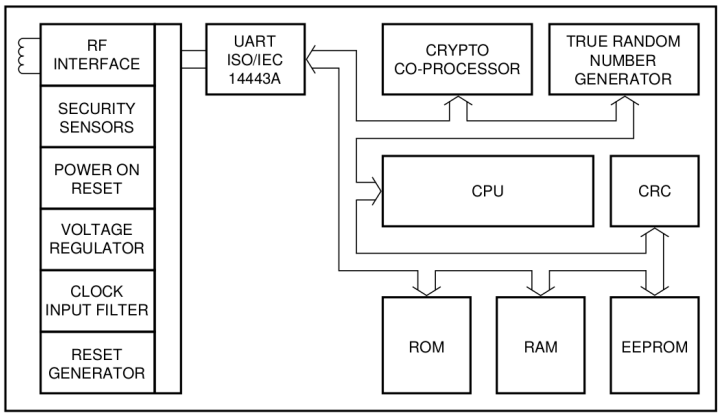

Figure 2.1: MIFARE DESFire EV1 Block Diagram [36]

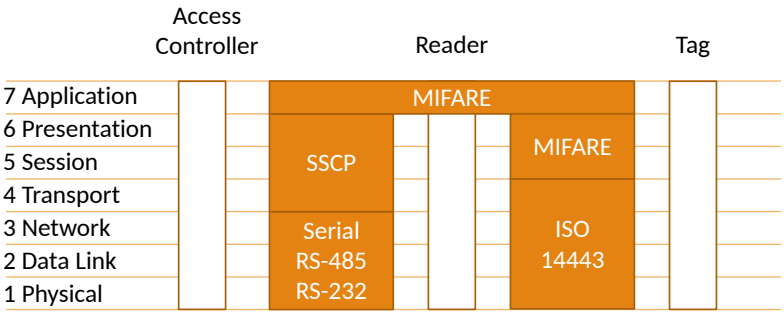

Figure 2.2: Parallel with the OSI network stack for a classical system

Reader The reader is the interface between Verifier and Prover. The reader is located at the border of the secured area and the unsecured world. In our case, we use the ARC-A from STid (see Appendix **??** for technical details). This device is compatible with SSCPV1 (for transparent mode, see chapter 3) and SSCPV2. It is also compliant with NFC specifications, that will allow to interact with smartphones.

Prover The Prover is the RFID token. It connects to the Reader in order to interact with the Verifier and authenticates itself via challenges and authentication mechanisms. In this use case, the Prover is a MIFARE DESFire EV1 RFID token.

Key manager The fourth entity is the key manager. This particular entity will generate the keys used for authentication. In some cases, this task is handled by the Verifier. However, for a better power separation and thus a better security (the key generation and management being a very important aspect, delegate it to an external party can be more secure), it is preferred to delegate it to another external entity.

The MIFARE DESFire EV1 card itself has an architecture as defined in Figure 2.1. We will focus on the right part of this diagram as it is the computational one.

2.2.2 Communication stack

We can make a parallel with the Open Systems Interconnection (OSI) network stack for communication with the Tag. Figure 2.2 shows this parallel and the different protocols taking place in the communication. We can see that the upper layers between the Access Controller and Reader are handled by SSCP for Presentation and Session. Indeed, a secured session is created between the two entities.

Between Tag and Reader, those layers are handled by the MIFARE protocol, meaning that a new session and security layer is taking place. Indeed, the reader has two secured sessions open (with the Access Controller and with the Tag). The reader thus performs cryptographic operations to maintain those sessions secured.

2.3 Prover: MIFARE DESFire EV1 Card

This section describes the technical capabilities of the MIFARE DESFire EV1 card. It also covers the exact role or the Prover in the system and some of its vulnerabilities.

2.3.1 Computational capabilities

As depicted in Figure 2.1, the MIFARE DESFire EV1 card embeds a CPU: This multi-purpose CPU handles the MIFARE commands and orchestrates the whole authentication process. The Crypto co-processor is used for every cryptographic operations. The MIFARE DESFire EV1 can handle multiple cryptographic algorithms [36]:

DES with key size 56/112/168 bits, used for legacy purpose.

3DES with key size 56/112/168 bits[1]

AES with key size 128 bits, recommended by NXP as it is more secure.

Those cipher blocks are used in a Cipher Block Chaining (CBC) fashion for encryption and in CBC-MAC when message authentication is needed. Communications with the reader can be in plain-text, plain-text with Message Authentication Code (MAC) or encrypted with a Cyclic Redundancy Check (CRC). The CRC is computed by a dedicated Hardware Peripheral embedded in the tag.

In this thesis, AES is used with encrypted communications. As DES is deprecated and AES is stronger in terms of security than 3DES, AES has been preferred to the other algorithms. The encryption key used for communication is the session key, generated at authentication (covered in Section 2.6).

2.3.2 Memory layout

The memory layout of the MIFARE DESFire EV1 is composed of multiple applications. Each application can contain files that are writable/readable only if the access to the containing application is authenticated. Different rules can be given to each application in order to segment the access rights. Each application can have multiple dedicated keys (for authentication, reading, writing) and each file can have specific access rules (write and read).

The files can have multiple types [36]:

- Standard data files

[1] 3DES is possible in 2K3DES or 3K3DES (two or 3 keys).

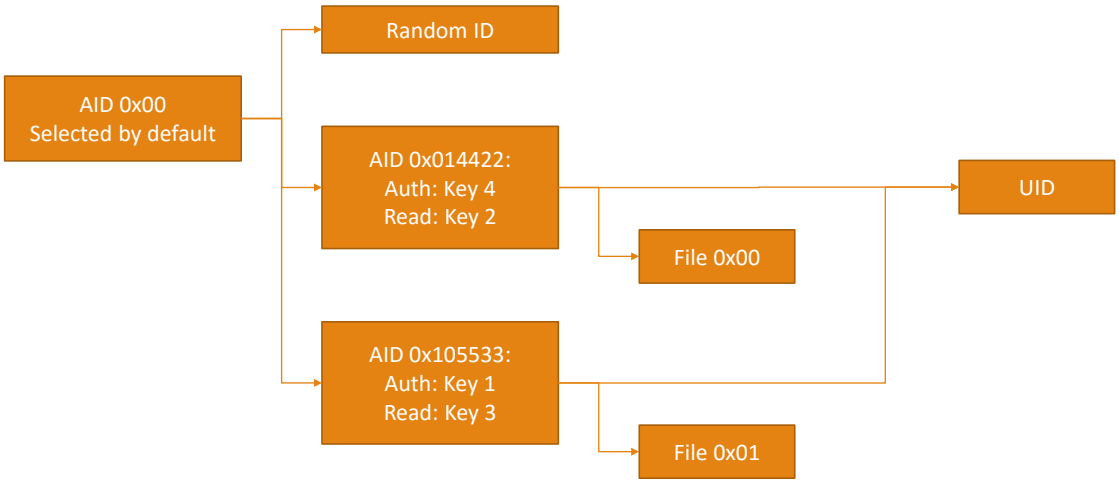

Figure 2.3: MIFARE DESFire EV1 Memory Layout

- Backup data files
- Value files with backup
- Linear record files with backup
- Cyclic record files with backup

In our case, only the standard data file is used. This file is a simple linear memory that can be read or written exactly as a standard memory.

Scheme 2.3 describes the memory layout as it is configured in the card used for this thesis. We will only use application $0x014422$ as we only need to authenticate the card. File $0x00$ of this application contains an ID and signature that has to be checked by the Verifier in order to authenticate the user.

2.3.3 UID and Random ID

The UID of the card (7 Bytes) is fixed at production and can be protected by a random UID. A random UID means that the card does not send the actual UID first but only a random UID for the session. The actual UID is accessible only after authentication with a specific application. This UID is stored in a special sector of the Electrically-Erasable Programmable Read-Only Memory (EEPROM) and is read-only.

The random UID is fixed for the session and is picked at the tag's initiation by the reader. It does not allow the reader to identity the card between two different sessions.

2.3.4 Key diversification

Keys used in the card are diversified, except for the authentication. This diversification takes the real UID of the card and a master key. Those are combined in order to generate a unique key per card. This technique allows to reduce the impact of a key leak. Only the card associated to the leaked key is compromised and not the whole system.

However, as the UID is unknown until authentication with the tag, the authentication key has to be unique for the whole group. The disclosure of this key hence compromises privacy of group-members as everyone can request the real card's UID.

2.3.5 Randomness

The MIFARE DESFire EV1 card embeds a true random generator [36]. This random generator is used for the authentication flow. It will generate *RndB* that is used to generate the session key. It is also responsible of the random UID. As this Random ID is generated via a truly random generator, we can say that the tag is not traceable uniquely via the random UID.

2.4 Reader: STid ARC-A

The reader is the STid ARC-A (the exact reference is ARCW33APH57AA1) which embeds a PN532 from NXP Semiconductors. The reader stores the keys and authenticates with the card. The family of this reader has been certified by ANSSI [4] and is hence compliant with the previously exposed recommendations.

2.4.1 Memory

The memory will contain the authentication keys of the reader. It can store several 128 bits keys stored by index in the memory. This memory is protected against violation via different protection mechanisms based on movement detection via accelerometers and voltage surveillance. In case of tampering, the memory is wiped to protect the contained data. Nevertheless, this memory is also the big weakness of the system as explained in Section 2.7.

2.4.2 Role in the authentication

The reader is used in SSCPV2 non-transparent mode (see [39] for the exact specifications). This mode requires an authentication between Access controller and Readers. Two keys are generated for each session (for signature and encryption). The reader is then responsible of the security for every communication with the Prover (the card). It also authenticates directly with the card. The reader is responsible of most of the cryptographic operations. It simply receives keys and commands from the Access Controller and sends the result back to it, encrypted or not depending on configuration used.

2.5 Access Controller: Scaline SC415

This section covers the Access controller and its role in the authentication process. The Access controller is a SC415 from NCS Scaline (see Appendix **??**).

2.5.1 Technical details

The SC415 embeds a Freescale/NXP Kinetis K61 CPU which has a 32 bits ARM architecture [35]. The clock frequency of the CPU is fixed at 180 MHz. This CPU embeds a cryptographic hardware peripheral that can perform DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms. The Kinetis K61 also has a truly random generator used as seed for a Pseudo Random Number Generator. In term of memory, the Access controller has 64 MB of RAM and 128 MB of Flash

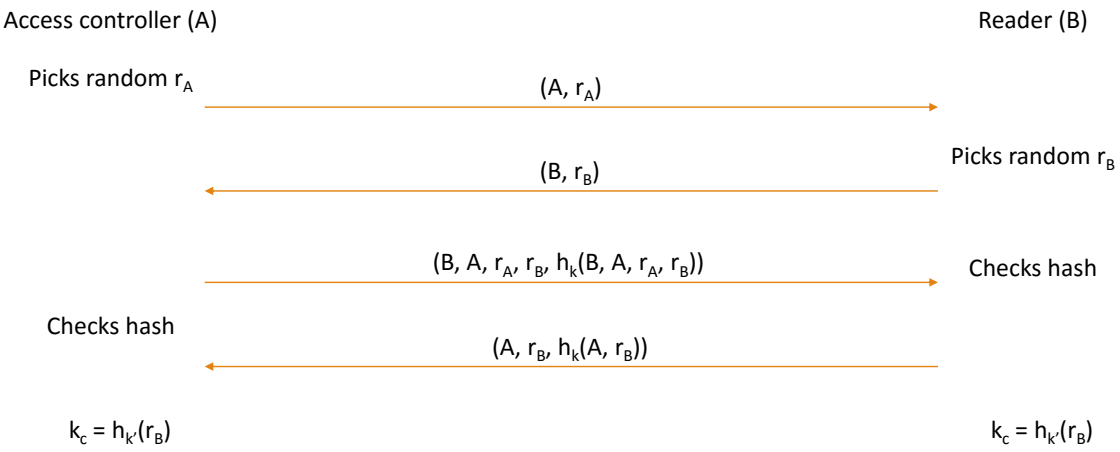

Figure 2.4: Authentication between Access Controller and Reader

memory. The Operating System (OS) is Linux based and can be accessed through serial interface and TFTP. It can be connected to IP network and the communication with the readers is done through serial interface in RS422 or RS485.

2.5.2 Role in the authentication

As seen in previous section, the Reader handles most of the cryptographic operations. The Access Controller only sends cryptographic keys and commands to the Reader that handles the rest. It will also get data from the tag and compares its signature with the database. The Access controller authenticates with the Reader, generates the session key and then simply sends commands and cryptographic keys used for authentication with the Tag. At the very end of the authentication flow, it also checks the signature of the read data and accepts or denies access.

The main work of the Access controller is to open the gate or the door to grant access.

2.6 Authentication scheme

The authentication scheme can be cut in four parts:

1. Authentication between Access Controller and Reader
2. Authentication between Reader and Card
3. Key diversification and read of signature
4. Signature check

2.6.1 Authentication between Access Controller and Reader

The authentication is described in [39]. This authentication is done between the Reader and Access controller. It generates session keys for ciphering (k_c) and signature (k_s which is derived from k_c). The authentication follows the ISO/IEC 9798-4 unilateral standard [21] and is represented in Figure 2.4.

The Access controller generates a random value r_A and sends (A, r_A) to the Reader. The Reader then chooses r_B and sends $(B, A, r_A, r_B, h_k(B, A, r_A, r_B))$ back to the Access Controller.

The Access controller computes $h_k(B, A, r_A, r_B)$[2] and compares it with the one received. If those match, it sends $(A, r_B, h_k(A, r_B))$ to the Reader. The Reader can then compare h_k received and the one computed locally. If those match, the session keys are computed based on r_B (typically $k_s = f(k', r_B)$ and $k_c = f_1(k', r_B)$ with $f()$ and $f_1()$ ciphering functions depending on a shared key k').

We can see that r_B is the only value used to generate the key. It is thus really important that the Reader generates true randomness for its generation. Also, keys k and k' that are used for authentication are known by both parties and thus also stored in the Reader's memory. Retrieving k can raise potential security issues as it would be possible to inject messages in the authentication flow. By retrieving k', the whole authentication becomes useless as k_s and k_c can be computed directly via r_B that transits in clear on the channel. It is really important to keep those two keys in a secured environment.

2.6.2 Authentication between Reader and Card

After authentication between Reader and Access controller, the Reader scans surrounding cards and selects the target. The card automatically opens application $0x000000$ which basically allows nothing. In this application, it is possible to get the list of applications of the card, read the Random UID[3] and authenticate with another application.

In order to get the real UID of the card, it is necessary to authenticate with an application. In our case, we authenticate with application $0x014422$. This authentication is described in [31] and its scheme is confidential. It is a proprietary three-pass mutual authentication based on random numbers with key agreement.

The authentication is based on AES in a CBC fashion and uses a non-diversified key k_{auth} that is thus common for every tags of the group. The Initiation vector (IV) is initialized to $0x00$ before each authentication. At the end of this authentication, the generated AES 128 *bits* key (k_{sess}) is used as session key for the rest of the session.

The final IV of each cryptographic operations is kept for the next operation. This allows having pure freshness of the IV for every encryption.

Key k_{auth} is common for every cards of the set, thus for the whole company that uses the authentication system. This key only allows to authenticate with the card and to get the real UID. Other keys are used to read and write to the memory of the card and those are diversified. This aspect is discussed in Section 2.7.

2.6.3 Key diversification and signature read

Key diversification allows to have different keys, depending on the card itself, for every tag of the group. In this case, the unique key is function of a master key, a diversifier and the real UID of the tag. Using diversification allows to be sure that it is nearly impossible to retrieve other keys of the group if one is compromised.

After authentication with the application, it is possible to get the real UID of the card. This UID is used for key diversification in order to retrieve the read key k_{read} that allows to read data from the application. This last key is then used in another authentication between the reader and the card in order to read the data from file $0x00$.

[2] $h_k()$ is a keyed hash function where a key is needed to hash a message or check the signature.

[3] Randomly selected at activation of the card from the reader and constant for the whole session.

This file contains random data, selected at the write of the card and the signature of those data using k_{sign}. Those data are used to authenticate the identity of the tag.

2.6.4 Signature check

Finally, the Access controller checks the signature of the data read from application $0x014422$ in file $0x00$. The signature key is the same for every tags. This file is composed of an ID for the tag and the signature of the concatenation of the UID and the ID: $D = ID|HMAC(k_s, ID|UID)$ where k_s is the signature key and the HMAC function is an HMAC-SHA256. If the signature matches, the access is granted and the door opens.

2.7 Weaknesses

We can identify some weaknesses of this system. The biggest issue is the storage of the keys. The authentication keys are stored directly in the reader which is, by definition, outside of the secured area. Whereas one of the key rules in security is to store keys in the most secured part of the system. This issue is covered in Chapter 3.

The second issue concerns privacy of the user and its anonymity. As discussed previously, there are multiple keys used in order to access the memory of the card. The card uses a random UID before authentication which is publicly available for any reader activating the tag. Thus, the only protection of the real identity of the tag is this authentication key, shared for the whole token set. As discussed previously, this key is also stored in the reader and so can be recovered via different techniques such as side channel attacks, memory tampering, communication interception,... The anonymity of the user in face of other users is thus compromised if this key is revealed.

But anonymity of the user against the Access controller is impossible with such a system. This means that technically, the Access controller can track every entries of employees in the company. The access control regulation for companies stipulates that the time measurement cannot be done by the same system as the Access controller. In the use case of a company access, this issue is not really important as the user accepts it. But if we use the same system for ticketing or public transportation, the privacy of the users is a lot more important as he can be tracked by the service provider itself. Today, a lot of people are concerned by their privacy and are fighting against this kind of privacy violation. This issue will be discussed in Chapter 4 where an alternative is described.

2.8 Conclusion

This part covered a classic implementation of an authentication system using MIFARE DESFire EV1 tags. This kind of tag is widely used in access control and transportation systems which is fine in specific use cases such as access authentication. However, it is necessary to identify the use case and the obligations related (e.g. privacy for public transportation).

We discussed the computational capabilities of the token and the different flaws that the system brings in term of security and privacy. Some of those issues can be solved via simple solutions that are covered in next parts of this thesis such as anonymity (see Chapter 4) and key storage in the secured area (see Chapter 3).

Chapter 3

Transparent mode (MIFARE DESFire EV1)

3.1 Introduction

As seen in the previous Chapter, the classic access control implementation brings some flaws in term of security and privacy. The majority of those issues are related to key storage in the Reader, which is thus interesting to investigate. This part covers an implementation of transparent mode on a system that was previously not transparent.

Transparent mode enables the Reader to simply act as a relay but requires the Access controller to manage every layers of the authentication. This means that the Reader does not do any cryptographic operation on the communications but simply relays messages to the token, it only handles the last layer of the communication stack related to RF communication.

3.2 System description

The transparent architecture is pretty similar to the classic architecture: Components are the same but those do not play the same role. The system still includes the same four previous entities. The difference is that the Verifier is now the Access controller and the Reader becomes a simple relay. Figure 3.1 shows the architecture used in transparent mode in front of the classic mode. The different colors indicate that the communication is intercepted and managed by the reader in classic mode. However, in Transparent mode, we can see that the communication goes through the Reader as it does not perform any operation related to security on the communication. We can point out that this architecture follows the ANSSI's recommendation for transparent mode (Figure 1.2 from Chapter 1).

3.2.1 Prover

The Prover (the tag) can act exactly the same way as in classic mode: It connects to the Reader which transmits every communication to the Access controller. The only difference is that the authentication is now performed by the Access controller. This change in the system is invisible for the Prover.

However, the communication delay can be slightly longer as messages now travel a greater distance (tag to Reader versus tag to Access controller). Also, the Access controller has different

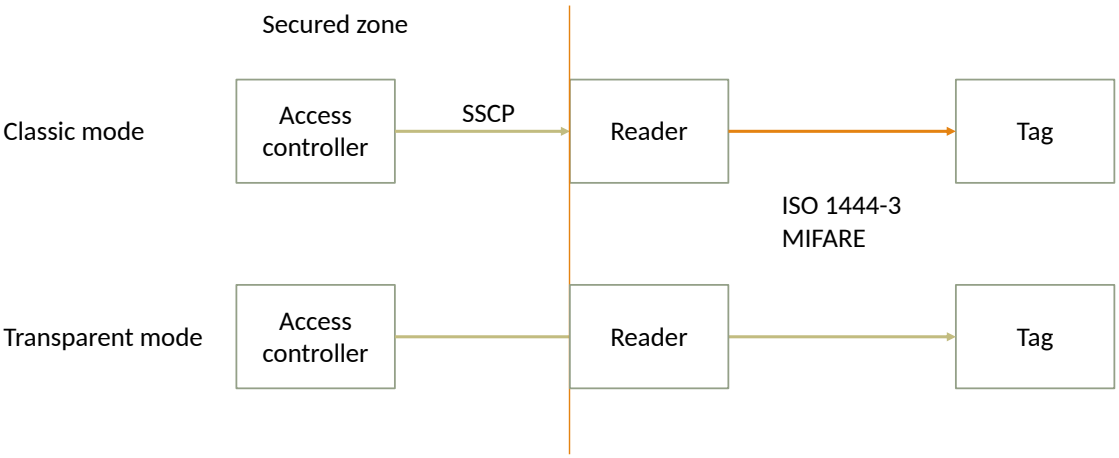

Figure 3.1: Transparent system

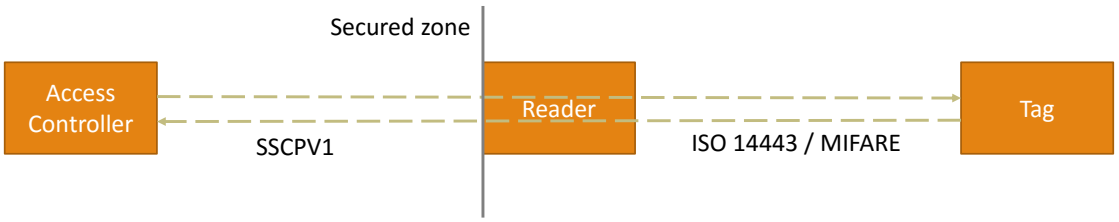

Figure 3.2: Transparent communication

operations to performs which can conduct to slower authentication than the reader which has only one operation to do.

In the other hand, a delay of 500 ms is allowed for authentication. Human does not notice if the delay between the tag scan and the door opening is less than 500 ms. ISO 14443 [19] also allows the tag and the Access controller to ask for more time to perform the computation. This time can go up to 300 ms which is sufficient to perform any operation of authentication.

3.2.2 Reader

The reader is now configured in transparent mode, using SSCPv1 [38]. SSCPv1 allows to communicate directly to the RFID token without the cryptographic layer of the reader via the Transceive function. The reader (in our case the STid ARC-A **??**) and the protocol have to be compatible with Transceive mode, which is the case of the PN532 [30]. The integrated memory of the reader is not used anymore as the keys are not stored in it.

Operations performed by the reader are now simpler than before:

1. Scan cards and activate the nearest token
2. Transmit commands from Access controller to the token
3. Responds to the requests from the Access controller

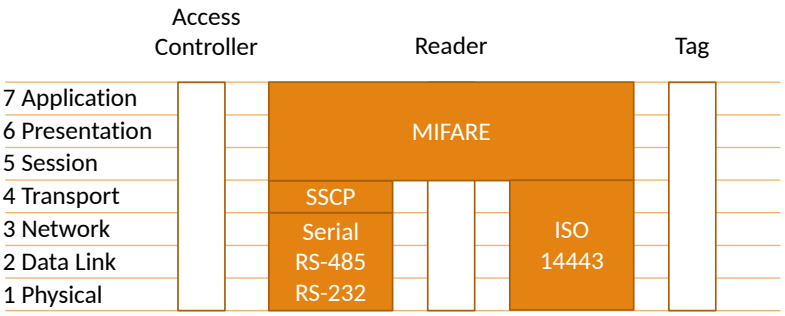

Figure 3.3: Parallel with the OSI network stack for a transparent system

The whole authentication process is detailed in Figure 3.2. We can see that the reader really acts as a relay and does not do anything on the frames sent and received. The implementation is also detailed in Section 3.4.

Letting the Reader encrypt the communications is pointless as keys are stored in its memory. This means that the encryption keys are available outside the secured area. It is hence possible to recover them by physical attacks as the reader is physically available. If those keys are known, the whole security of the communication is compromised. This issue is the biggest argument for the Transparent mode.

3.2.3 Access controller

The biggest changes made are done on the Access controller part. This entity now handles everything from the command layer to the cryptographic one (see Figure 3.3 versus Figure 2.2). This means that the Access controller now performs more operations and is thus more solicited than before. The NCS SC415 embeds a cryptographic co-processor [28] that eases the transition to transparent mode.

3.2.4 Communication stack

The communication stack described in previous section (Figure 2.2) is deeply modified as the Access Controller now handles almost everything in the communication stack. Figure 3.3 shows the new network stack ads it is in transparent mode. It be pointed out that only one security session is needed in this scheme (the MIFARE session) whereas in the classic scheme, two where used (between Reader and Access Controller and between the Reader and Tag). This change in security means that the overall computational complexity decreases as the Reader now only transmit messages without touching layers above 4.

Some can think that using only one security layer is less secure. This assumption is wrong because of the parties involved in the security layers. In classic mode, the Reader is involved in the two secured sessions, it manages cryptographic keys and stores them. The Reader is located outside the secure zone, meaning that it is potentially attackable via different means. It can thus compromise the whole security of the communication.

As in transparent mode keys are not shared with the Reader, but only between the Access controller and the Tag, those keys always stay in the secured area which by definition is more secured than the Reader. This means that key recovery is more complicated than before and thus that the system can be considered more secure. Figure 3.3 shows the role of the Reader which is simply to handle lower layers.

Protocol	Ciphering	Signature
SSCP	AES (CBC)	HMAC-SHA1
MIFARE	AES (CBC)	CBC-MAC

Table 3.1: Comparison of security for SSCP and MIFARE protocols

3.3 Differences with the classic implementation

This section summarizes the gains obtained by using the transparent mode instead of the classic system described in Chapter 2. Those changes are detailed in term of security but also in term of the communication stack.

3.3.1 Security

As seen in Figure 3.3, only one security layer is needed for this system. This means that there are less cryptographic operations performed per message. The difference is that in the classic implementation, the removed security layer (security with the reader) was handled by the Access controller and the security with the tag was handled by the Reader. Now, the Reader does not perform any security operations: We can thus use cheaper readers that only handle the physical layer of the communication stack.

On the other hand, operations performed in those two stacks do not bear the same complexity. In the MIFARE protocol, every communication has to be CBC-MACed in order to check the authenticity of messages. In the SSCP protocol, messages are authenticated using HMAC-SHA1. Also, every secured message is encrypted using AES in CBC in both cases. Table 3.1 summarizes those differences and algorithms used.

Key sizes are the same (128 *bits* for both) and the key generation follows pretty the same key agreement schemes, the only difference is the randomness used. The PN532 as no dedicated hardware random generator [30] and no information can be found about an external random generator. However, we know that the MIFARE DESFire EV1 and the SC415 both embeds a truly random generators [36] [35].

3.3.2 Communication

The Access controller now handles one communication layer more than in the classic implementation. This layer is pretty simple as the MIFARE communication protocol is command based [34] and does not bring much more complexity. We can say that the difference is thus really minimalistic in term of computational footprint.

The Access controller performs encapsulation for every message, as described in Figure 3.4. Every command to the reader is simply prepend by the InDataExchange command ($0x40$) and the card number on one Byte [30]. The card number correspond to the index of the card scanned by SSCP_TranspScanRaw (see Section 3.4). Every successful message is replied by an answer preceded by the code $0x41$. In case of error, the status is different of zero and depends on the exact error [30].

In order to directly address the PN532, every command passed to the reader uses the SSCP_Transceive function ($0x0014$) [38].

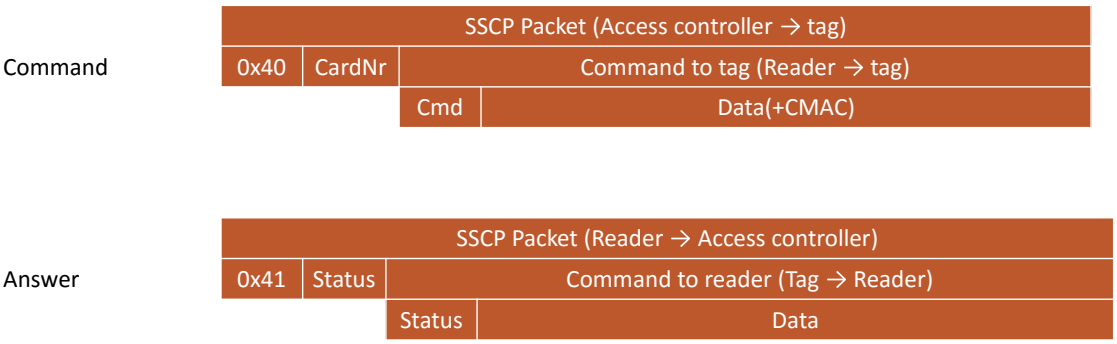

Figure 3.4: Transparent commands encapsulation

3.4 Upgrade of the system: Implementation

In term of implementation, the library is upgraded to use Transparent mode and the SSCP_Transceive function ($0x0014$). Every "classic" function is re-writed in order to use this transceive function and handle the computations previously done by the Reader.

3.4.1 Key exchange and authentication

The authentication with the tag is now completely handled by the Access controller. Thus it has to be re-implemented. The whole process can be represented as the following sequence of functions. Those functions sends a command per function to the reader that simply relays them to the tag. Every message sent to the token is CBC-MACed except the authentication messages.

SSCP_TranspScanRaw

Activates the surrounding tags and selects the first one for authentication. The reader completely handles this command because it only involves RF communications. This function starts the magnetic field of the reader in order to power up the surrounding tags. Activated tags respond with their UID (or random UID)[1]. The tags are initialized and ready for authentication. By default, the first detected tag is used for authentication.

SSCP_TranspMFDFGetApplications

Gets the list of applications configured on the tag. Each available Application IDentificator (AID) is returned. This listing is interesting in order to recognize tags used for authentication. We can directly stop the authentication if application $0x014422$ is not listed here as it means that the token is not compatible with our authentication system. We can thus retry with the next detected tag from previous step.

SSCP_TranspMFDFSelectApplication

Selects the application by AID to authenticate with. In our authentication scheme, we authenticate with application $0x014422$. By default the selected application is the root application (AID $0x000000$) which only allow to access the application list and the card's random UID.

[1]ISO 14443 defines an anti-collision mechanism that is used here but this is only related to RF communication so we will not dig into details [19].

SSCP_TranspMFDFLoadKey

Loads the authentication key to the dedicated memory. The goal of this function is to keep the same structure as the non-transparent library and to give an index to each stored key. Normally, this index correspond to the memory space of the Reader. In transparent mode, this index is simply the index of the storage array. This function simply copies the key to a C static array at that specific index.

SSCP_TranspMFDFAuthenticate

This function authenticates with the previously selected application using the key loaded to the key memory. Details of this authentication are confidential and detailed in [31]. When the application is authenticated, it becomes possible to access its functionalities such as accessing the real UID.

This function generates a new session key for the whole communication. Normally this session key would have been stored in the Reader and never known by the Access controller.

Before authentication, the IV is also reset to zero. It is important to note as it propagates for every other operation done to the card.

SSCP_TranspMFDFGetCardUID

Gets the real UID of the tag. This function reads the UID after authentication, it is needed as the tag uses random UID for privacy. The tag is configured to allow access to the real UID only after authentication with an application. It is a solution to enhance privacy of the user.

genDiversifiedKeyDESFireAES

Generates the diversified reading key for the tag. This function takes the master reading key of the application and the real UID of the card. It outputs the corresponding diversified key corresponding to this particular tag. The exact diversification algorithm is detailed in Reference [31]. The diversification takes the master key and the real UID of the card as parameter.

SSCP_TranspMFDFLoadKey

Loads the diversified read key in order to read application $0x014455$ and the signature contained in file $0x01$. Again, this function is only there to keep the same structure as the non-transparent library.

It would also have been possible to load every keys once at the initialisation of the whole system if we did not use diversified keys. But this could have worsen the security of the system as every key would have been stored for a longer time in the Reader. If we only load the key when needed, the risk of malicious key recovery is smaller.

SSCP_TranspMFDFAuthenticate

Authenticates with the reading key of the selected application in order to have access to its content. This function is the same as before, the difference is the key used and t fact that the selected application is now $0x014455$. Again, the IV is reset and a new session key is generated. This session key is used for the next communications.

SSCP_TranspMFDFReadData

Reads the content of the file $0x01$ of the previously selected application. As the Access controller is authenticated with the application, access to the data is granted and it becomes possible to read the signature.

The data contained in the file are composed of some bytes of data and a signature. This signature is the signature of the first bytes signed with another diversified key. The signature is checked by the Access controller.

SSCP_hmac

Computes locally the signature of the previously read data. This computation is done by the cryptographic peripheral with the signature key. As said previously, this key is diversified using the same algorithm as for the authentication key (see Reference [31]). As the function names suggests, this signature is an HMAC of the data. This HMAC is in fact a CBC-MAC relying on AES.

If the signature holds, access is granted and the Access controller opens the door. For this thesis, the lights of the Readers where lighten in green whereas in case of an access denied, those change to red.

3.4.2 Key management (memory)

Keys were previously sent to the Reader's protected memory. Now that the protocol is transparent, those keys have to stay in the Access Controller. The Transparent library thus includes a dedicated memory, the same size as the reader's memory that enables to store the same amount of keys and that can be accessed the same way.

The keys are stored in a C static 2D array that is zeroized[2] when not used anymore. This enables to be sure that this memory cannot leaks keys when reused in another process.

3.4.3 Encryption

Encryption of the communications is now handled by the Access controller. It embeds a dedicated hardware accelerator that can handle this encryption/decryption. It is important, in case of big chunk of data, to have such an accelerator as otherwise, the CPU can fastly be overwhelmed.

The session key is agreed at authentication with the tag. This key is used for encryption of each message and answer between the Access controller and the tag. The ciphering is an AES CBC ciphering which requires an IV. This IV is propagated from the previous to the next enciphering or MACing operation. This IV is reset at each authentication so that tag and access controller can always have a synchronized IV.

3.4.4 HMAC

Each communication is authenticated by an HMAC. This HMAC is a CBC-MAC using the session key agreed at the authentication step. This key is the same as the one used for the enciphering. The HMAC uses the IV from the previous cryptographic operation and is also updated at the end of the HMAC.

This HMAC was not previously implemented in the Access controller's library, its implementation uses the cryptographic hardware accelerator as ciphering block and is tested against the NIST 800 specification [15]. This specification is the CBC-MAC implementation, it regroups

[2]Every byte is set to zero

test vectors that covers the most important inputs in order to cover the whole possibilities of the CBC-MAC and check its correct implementation.

3.5 Weaknesses

The transparent system now protects the keys as those stay in the Access controller. On the other hand, it adds delays to the authentication process and can create visible lags for the user. The Access controller has to present a sufficient computational capability in order to handle the requests as fast as possible.

The privacy issue is still present with a transparent system. Indeed, the Access Controller still has access to the identity of the tag. Moreover, the authentication key is still shared for every tokens. The key stays in the secured area (in the Access controller) so the attack surface decreased in comparison with the classic system.

In a relatively small company this anonymity issue is not really important as the probability of discovering the authentication key is small and the gain is also small as every employee knows each other. But in the case of public transport, ticketing etc. In those use case, better schemes have to be used, those are explained in the next sections. Also, those usages need a protection of the user's privacy as it is not needed to know its identity.

Attacks such as Relay attacks are always possible in transparent mode. However, it can be more complicated to perform as the overall communication is now slightly longer. This therefore decreases the freedom of an attacker but it is slightly negligible.

3.6 Conclusion

This section exposed the improvements made to the system in order to use the reader as a simple relay. The Access controller now communicates directly to the tag without any operation from the Reader except for the physical (RF) layer. Every command to the tag is thus encapsulated using the Transceive function of SSCPv1. This improvement is very important for the security especially regarding the key storage as they are now kept in the secured area. This modification displaces some computations from the Reader to the Access controller but does not bring any issue as the delay allowed between two messages is very important for ISO 14443 compliant devices [19].

This new implementation does not bring anything in term of anonymity with respect to the Access Controller. The user still has to be identified in order to authenticate the access. However, we can point out that the gain in security for the authentication key can help to keep the anonymity of the users against an external attacker. Indeed, recovering the authentication key is not possible in the non-secured area thus the critical point, which was the Reader, is now useless for an attacker.

Chapter 4

Anonymity: Group signature

4.1 Introduction

Looking at previous sections, we see that systems relying on MIFARE DESFire EV1 tokens are good for access authentication in companies or small organisations. But today, a lot of other use cases are proposed for RFID tokens, for example in public transportations. Those new use cases bring a lot of new problems like bigger groups and protection of the privacy of the user.

In this chapter, group signatures are covered. Those allow to sign messages for a group. Meaning that the Verifier can only say that the key used to sign the message was part of the authorized group but it is impossible to recover exactly which key signed the message. This new theory adds a lot of possibilities for anonymity in access control. This kind of schemes allow anonymity in access control applications such as public transportations, event ticketing, etc. In this chapter and the remainder of this thesis, we will use a different use case: We will take into account a subscription to a concert hall. This use case is further detailed in this chapter.

4.2 Theoretical background

Chaum and Van Heyst [8] introduced settings of group signature the following way:

> There is a group with several members that can sign messages. Each group has a single group manager which has the ability to generate new members to the group but also retrieve the identity behind a signature. The group manager has a secret key $gmsk$. Each signature can be checked via a the group public key gpk and each member of the group has a secret and personal key $gsk[i]$ that allows him to sign messages.

Each signature σ generated by a key $gsk[i]$ can be checked using gpk and the identity of the signer can be revealed by the group manager with key $gmsk$. In the other hand, without the $gmsk$ key, the anonymity of the signer must stay complete and no information about it has to be revealed. This ensures a total anonymity of the signer and thus a non-traceability of the signatures.

4.2.1 Formal definitions

When speaking of group signature, there are some key definitions that have to be known. Those will be used in every group signature scheme of this thesis and are defined in [6] and [37].

Full-anonymity

An adversary produces a message m and a pair of group-member identities gsk_1 and gsk_2. He signs the message $\sigma = sign(gsk_i, m)$ using one of the two identities at random. The adversary must have no advantage over a probability of one half to determine the identity that signed the message.

Using this definition, a strong adversary is an adversary that can corrupt every group members via the acquisition of the group-member's secret keys $gsk[i]$.

Full-traceability

This definition claims that group members that colludes and shares multiple $gsk[i]$ keys cannot generate a signature σ that will not be linked to a $gsk[j]$ of the colluding group. In other word, this means that a signature is always linked to the signer and that its identity is always reachable using the group management key $gmsk$.

This definition is also valid if the colluding group knows the group management key that opens the identity behind a signature $gmsk$.

Unforgeability

An adversary is unable to produce a signature σ of a message m that will be valid for the group if he does not have any valid key $gsk[i]$. This definition is really important. Indeed, if an adversary was able to produce a valid signature without being part of the group, the whole scheme would be useless.

Unlikeability

This property is related to anonymity and full-traceability. Given a set of signatures from different group members, it must be difficult to deduce a link between two signatures. Hence, two signatures from the same member have to be undistinguishable from signatures of other group members. Furthermore, two signatures of the same message, signed by different group members cannot be linked either.

The definition is more complete in [6] but this formalism is not needed to understand the underlying principles of the different schemes.

Framing-proof

A framing attack is an attack where an adversary produces a signature that can be attributed to an honest group member. The hypothesis is that the adversary does not have access to previous signatures generated by the group-member and also that he does not have the $gsk[i]$ key of this group-member. Once again, if this definition is not respected for the scheme, the scheme becomes pointless as it becomes possible to sign on behalf of a non-revoked user for example.

4.3 Pairing based cryptography

Different cryptographic schemes exist for group signatures. Those are based on zero knowledge protocols and use asymmetric cryptography. In this thesis, we will focus on specific constructions that involve pairing functions and Elliptic Curves Cryptography (ECC). It is thus important to define Pairing Based Cryptography (PBC) and the implications behind those constructions.

RSA key-size	ECC key-size
1024	160 (18%)
3072	256 (8%)
15360	512 (3%)

Table 4.1: Key sizes in bits for ECC and RSA

The choice of PBC and ECC for group signatures is motivated by the computational power available that is growing nowadays as smartphones reach the cap of 2 GHz on multi core CPUs and memory grows to up to 6 GB of RAM [7]. However, the communication channels do not evolve as fast as the power capabilities and thus reducing signatures and key sizes via the use of ECC can be interesting. Indeed, elements and key in ECC compared to the same security for RSA are very smaller as depicted in Table 4.1 [26] [40]. We also see that as the key size increases (and thus the security), the relative gain in size also increases.

Those two criteria are currently driving the adoption of ECC. However, some curves used in ECC are subject to patents. Moreover, some concerns about the security of ECC are also expressed as it can be more vulnerable than RSA to quantum attacks such as the Shor's algorithm [29].

4.3.1 Elliptic Curves Cryptography

It is important to understand the basis of ECC and more generally Elliptic Curves in order to understand the schemes exposed in this chapter. [9] gives a great introduction to elliptic curves operations which is summarized here.

Put simply, an Elliptic Curve is an abstract type of group[1].

Elliptic curves are also depending on a Finite field $K \in \mathbb{F}_q$. For a general field K, we can define an elliptic curve E as

$$E : a_0y^2 + a_1xy + a_3y = a_5x^3 + a_2x^2 + a_4x + a_6, \tag{4.1}$$

where $a_0, a_1...a_6 \in K$. This equation is the general Weierstrass equation for elliptic curves. The elements of the group generated by E are points (x, y) located on the curve. In practice, this equation can be simplified as

$$E' : y^2 = x^3 + ax + b, \tag{4.2}$$

if the field characteristic of K is not 2 or 3[2]. This equation is called the Short Weistrass Equation.

In practice, we work over large prime fields where the Short Weierstrass Equation (4.2) covers all possible isomorphism classes of elliptic curves, so the curves we use will always be an instance of Equation (4.2). This means that K is the group of large prime numbers and thus that a and b are big prime numbers.

However, not every $(a, b) \in K^2$ are such that (4.2) is an elliptic curve. If there exists $P = (x_P, y_P)$ on $f(x, y) \equiv y^2 - (x^3 + ax + b) = 0$ and $\frac{\delta f}{\delta x} = 0 = \frac{\delta f}{\delta y}$ at P, then P is called a singular point and f is also called singular. Conversely, if no such point exists, f is non-singular, or smooth, and is then an elliptic curve.

[1] A group, mathematically, is a set of elements and a mathematical operation that can combine two elements in another of the group. This operation has to holds four properties: invertibility, associativity, identity and closure. For example, the integers with the addition.

[2] The exact demonstration of this simplification can be found at page 6 of [9]

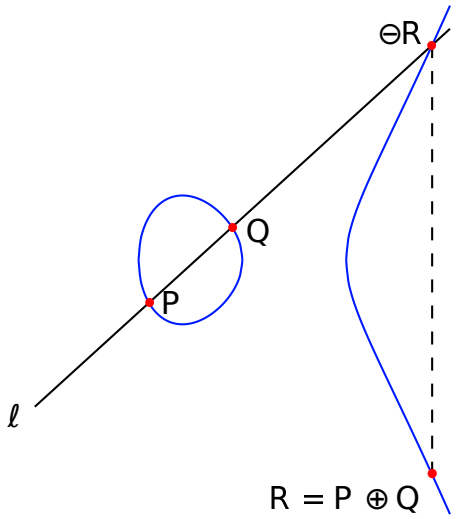

Figure 4.1: Elliptic curve addition [9]

Group law: chord-and-tangent

As said previously, a mathematical operation for the group has to be defined. In Elliptic curves, this operation is the chord and tangent operation [9]. This operation uses the fact that, over any field, a line intersects a cubic curve (a degree three equation in x and y) in three places. For understanding, the $\mathbb{R}$ field is used for explanations as it is more common. But in practice every field can be used.

The operation $R = P \oplus Q$ where P, Q, R are on the curve E is defined as follow: Each point of the group (points of E) is defined by its coordinates $P = x_p, y_p$, $Q = x_q, y_q$ and $R = x_r, y_r$. We create a function of degree one (a line, l) including P and Q. We take the third intersection of l with the elliptic curve E, this intersection is named $\ominus R$. This later point is "flipped" around the y axis ($y_R = -y_{\ominus R}$). This operation is shown by Figure 4.1.

The doubling operation $R = P \oplus P$ is performed using l as the tangeant of P. $\ominus R$ is computed as previously (the intersection of l and E) and R is also the flip of $\ominus R$. Figure 4.2 shows graphically this operation.

A new point has to be defined: $\mathcal{O}$ which is at the infinity. Intuitively, we can think of $\mathcal{O}$ being located infinitely high in the x coordinates and infinitely low in y (in the upper left corner).

$\mathcal{O}$ plays the role of identity for the group but it is also the only solution to operations involving a point and its inverse. $S = R \oplus (\ominus R) = \mathcal{O}$, the line intersecting R and $\ominus R$ intersects E in $\mathcal{O}$.

Discrete Logarithm problem

The Discrete Logarithm problem for a group $\mathbb{Z}_p^*$ goes as follow: Given $r, q \in \mathbb{Z}_p^*$ and p a prime number, find k such that $r = q \cdot k \mod p$.

For ECC, the problem becomes: Given two points P and Q of the curve, find s such that $sP = Q$ where s is called the discrete logarithm of Q in base P.

As this discrete logarithm problem is difficult in the elliptic curves group, it hence enables to construct asymmetric cryptography schemes using elliptic curves.

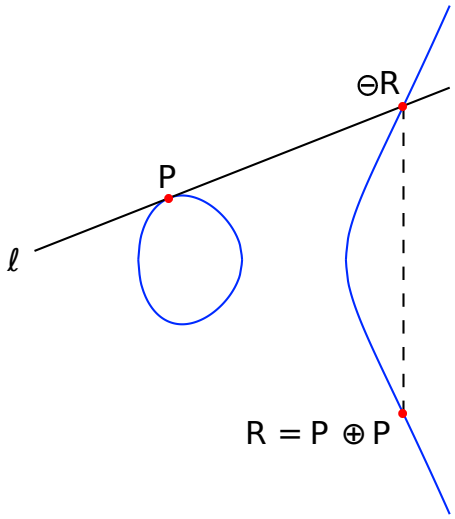

Figure 4.2: Elliptic curve doubling [9]

4.3.2 Pairing functions

This thesis will not dig into details of the pairing functions and their implementation as those are really out of the scope. [9] covers the basis of pairing function and their usage in cryptography. [23] and [27] are more focussed on pairing implementations and optimizations.

This thesis simply uses pairing as a black box and focusses on technical choices for the scheme to use for authentication. Hence it is not necessary to dig into the very details of pairing functions and a simple introduction to those is sufficient. The library used for the Proof of Concept (POC) in Chapter 5 is libPBC [25] which is written by Ben Lynn and available under LGPL licence. This library does not need any strong knowledge of pairing functions. The library implements Wail and Tate pairing functions which are really efficient on elliptic curves and are used in ECC [9].

Pairing definition

As stated in [9], a pairing is a bilinear map on a group M (or two) taking values in another group R:

$$< \cdot, \cdot > : M \times M \to R.$$

More generally, the pairing function is noted as

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

where $\mathbb{G}_1, \mathbb{G}_2$ can be defined in $E(\mathbb{F}_{q^k})$. $\mathbb{G}_T$ is defined in the multiplicative group $\mathbb{F}^*_{||}$.
We can thus compute e for $P, P' \in \mathbb{G}_1$ and $Q, Q' \in \mathbb{G}_2$:

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q) \quad (4.3)$$
$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q') \quad (4.4)$$

which follows, for $a, b \in \mathbb{Z}$:

$$e(aP, bQ) = e(P, bQ)^a = e(aP, Q)^b = e(P, Q)^{ab}. \tag{4.5}$$

Via 4.3, 4.4 and 4.5, the bilinearity property of the pairing functions can immediately be seen. It is the main property to understand for the different schemes illustrated in this thesis. The computation of those pairing functions is not really important to understand this thesis but more information can be found in [9], [23] and [27]. Also, as libPBC is open-source, it can be interesting to look at its source-code [25] for implementation details.

4.3.3 Strong-Diffie-Hellman

The different schemes exposed in this thesis rely on the Strong Diffie Hellman (SDH) assumption. It is hence important to define it properly.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups with g_1 generator of $\mathbb{G}_1$ and g_2 generator of $\mathbb{G}_2$. Consider the q-Strong-Diffie-Hellman problem:

The q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}..., g_2^{\gamma^q})$ as input, output a pair $(g_1^{\frac{1}{\gamma+x}}, x)$ where $x \in \mathbb{Z}_q^*$. An adversary $\mathcal{A}$ has advantage ϵ in solving q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$Pr\left[\mathcal{A}(g_1, g_2, g_2^{\gamma}, ..., g_2^{\gamma^q}) = (g_1^{\frac{1}{\gamma+x}}, x)\right] \geq \epsilon,$$

where the probability is over the random choice of g_2, with $g_1 = \varphi(g_2)$, of $\gamma \in \mathbb{Z}_p^*$ and of the random bits of $\mathcal{A}$ [12].

We say that the (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. In this definition, $\varphi(\cdot)$ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

4.4 Zero-Knowledge Protocols

Zero Knowledge Protocols (ZKP) are schemes that allow to protect the Prover's identity [24]. The Verifier should not learn anything but the fact that the Prover is part of the same group. Schemes presented in this chapter use ZKP heavily as those allow the signature of a message without giving any information about the signer.

Such protocols have two important properties:

Completeness If the Prover (P) really has the secret, it must be unlikely that it cannot convince the Verifier (V) of it.

Soundness If the Prover does not have the secret, the probability to convince V must be negligible. Even if Prover is unbounded.

Those two properties can be discussed regarding the Schnorr's Protocol [24]:

Let $\mathbb{G}$ be a group of prime order p with generator g. P proves the knowledge of g^x to V:

1. P chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and sends g^r
2. V challenges P via $e \xleftarrow{\$} \mathbb{Z}_p^*$
3. P responds with $f = r + e \cdot x \mod p$
4. V accepts if $g^f = g^r \cdot (g^x)^e$

In this scheme, P proves that he knows g^x without disclosing anything secret.

4.5 Schemes comparison

This section describes different Group Signature schemes based on Pairing Cryptography. Every scheme is defined first mathematically and then discussed with respect to the needs for the use case. For memory, the use case chosen for this thesis is an access authentication system intended to be used in public events such as concerts or sport events. It is necessary to keep the anonymity of the user (the Prover) but we want to be able to revoke this anonymity in case of accident such as a terrorist attack. The scheme will be implemented on smartphones. Verifiers are multiple Access controllers that are synchronized but not perfectly (delays can occur).

4.5.1 BLS (Boneh–Lynn–Shacham)

As an introduction to signature schemes, it can be interesting to first detail a simple public signature scheme relying on pairing. This scheme is presented in [11]. This construction is not a group signature scheme but enables to understand how pairing functions are used in order to build such schemes.

Key generation

The key generation is simply an exponentiation of a group element[3]:

$$\begin{aligned} g &\xleftarrow{\$} \mathbb{G} \\ x &\xleftarrow{\$} \mathbb{Z}_r^* \\ k_p &= (g, g^x) \\ k_s &= x \end{aligned}$$

Signature

For signing, a new group member is generated using an Hash function to map the message m into the cyclic group $\mathbb{G}$. The signature is hence as

$$\sigma = h^x = (Hash(m))^x .$$

This scheme relies on the q-SDH assumption.

Verification

The verification uses the pairing function in order to "exponentiate" the hashed value h without knowing x, the secret key:

$$\begin{aligned} e(\sigma, g) &\stackrel{?}{=} e(Hash(m), g^x) \\ \Leftrightarrow e(h^x, g) &\stackrel{?}{=} e(h', g^x) \\ \Leftrightarrow e(h, g)^x &\stackrel{?}{=} e(h', g)^x \end{aligned}$$

We can see that in this verification, the pairing function is used to "displace" the exponent. The properties of the pairing functions are used this way in every scheme presented below.

[3]Notation $y \xleftarrow{\$} \mathbb{Z}_r^*$ means that y is random picked in $\mathbb{Z}_r^*$

4.5.2 BB (Boneh-Boyen short signatures)

This scheme, presented in [13], uses the SDH assumption and no Random Oracle. This scheme is not initially a group signature scheme but enables to understand the different operations involved in the next schemes. Furthermore, it can be adapted to allow anonymous signatures. It is also interesting as it is the basis of the others schemes presented in this thesis. The interesting part of this scheme is the key generation which is very similar to the ones of group signature schemes presented below.

Key generation

The first operation is the key generation. The public key k_p and k_s are generated as follow

$$\begin{aligned} g_2 &\xleftarrow{\$} \mathbb{G}_2 \\ g_1 &= \varphi(g_2) \\ x, y &\xleftarrow{\$} \mathbb{Z}_p^* \\ u &= g_2^x \\ v &= g_2^y \\ z &= e(g_1, g_2) \\ k_p &= (g_1, g_2, u, v, z) \\ k_s &= (x, y) \end{aligned} \tag{4.6}$$

Where φ in 4.6 is an isomorphism from $\mathbb{G}_2 \to \mathbb{G}_1$. The public key k_p is then transmitted to every group members. The secret key k_s is used to sign messages and is thus sent only to the Prover.

A possible adaptation to anonymous signing is to send u and v with the signature and not the public key. In that case, the signature can still be checked and the privacy is kept. However, this also means that anyone can join the group simply by picking $x, y \xleftarrow{\$} \mathbb{Z}_p^*$. The only solution to avoid this issue is keeping g_2 secret but then Verifiers are not able to check signatures without it, unless we define a secret key for the whole group which contains g_2. Those concerns are at the basis of schemes such as VLR and BBS, presented below.

Signature

The signature simply consists of the computation of

$$\sigma = g_1^{\frac{1}{x+m+yr}}$$

where $m \in \mathbb{Z}_r^*$ and $r \xleftarrow{\$} \mathbb{Z}_p^*$. It is also really important to be sure that $x + m + yr \neq 0$. The full transmitted signature to the Verifier is (σ, r).

Verification

Verification is performed via the public key gpk. Verifier gets a message m and its signature σ as input. Pairing function is used to recover z which is part of the public key.

$$e(\sigma, u g_2^m v^r) = e(g_1^{\frac{1}{x+m+yr}}, g_2^x g_2^m g_2^{yr}) = e(g_1, g_2)^{\frac{x+m+yr}{x+m+yr}} = z$$

If this equality holds, the signature is verified and the access can be granted.

Specificities

As said previously, this scheme is not a group signature scheme but only a public key signature. It is thus impossible to use it for this thesis without adaptations. We can point out that this scheme never uses hash function and no random oracle (as it is defined in the cryptographic Standard Model).

However $m \in \mathbb{Z}_p$ can be the result of an hash function. It is the case in practice because the hash function can map messages to a defined set such as $\mathbb{Z}_p$.

In term of computations, this scheme only uses one pairing operation per verification and one per signature. It is hence really efficient for those online[4] operations.

The signature is really short as it is only composed of an element of $\mathbb{G}_T$ and an integer. This last specificity explains the interest of this kind of scheme for asymmetric cryptography.

4.5.3 BBS (Boneh-Boyen-Shacham short group signatures)

BBS is a group signature protocol [12] that also relies on SDH. It provides anonymity for signers as Chaum and Heist described it [8]. This scheme was firstly designed for communication between vehicles and is thus focussed on bandwidth and signature size: under 250 *Bytes*.

Key generation

The key generations takes n, the size of the group as input and generates the group's public key gpk and the secret keys $gsk[i]$ for $i \in [0, n-1]$.

$$\begin{aligned} g_2 &\xleftarrow{\$} \mathbb{G}_2 \\ g_1 &= \varphi(g_2) \\ h &\xleftarrow{\$} \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\} \\ \xi_1, \xi_2 &\xleftarrow{\$} \mathbb{Z}_r^* \\ u &= h^{\frac{1}{\xi_1}} \\ v &= h^{\frac{1}{\xi_2}} \\ \gamma &\xleftarrow{\$} \mathbb{Z}_r^* \\ \omega &= g_2^{\gamma} \\ gpk &= (g_1, g_2, h, u, v, \omega) \end{aligned} \tag{4.7}$$

The management-key is $gmsk = (\xi_1, \xi_2)$ which stays private to any party except the group manager that generates the keys. This key is needed to revoke anonymity and identify a signer. γ is also private and stays the private group key.

The group-member's secret keys are generated for each member ($\forall i \in [0, n-1]$) using the following algorithm:

$$\begin{aligned} x_i &\xleftarrow{\$} \mathbb{Z}_r^* \\ A_i &= g_1^{\frac{1}{\gamma + x_i}} \\ gsk[i] &= (x_i, A_i) \end{aligned}$$

Each $gsk[i]$ is sent securely to the group-member.

[4]Online operations are the one performed in real-time by the system. It is the opposite of offline operations that can be precomputed before the action of the user.

Signature

Given the public key gpk, the message to sign M and the user's private key $gsk[i]$, computes:

$$\begin{aligned} \alpha, \beta &\xleftarrow{\$} \mathbb{Z}_r^* \\ T_1 &= u^{\alpha} \\ T_2 &= v^{\beta} \\ T_3 &= A_i h^{\alpha+\beta} \\ \delta_1 &= x_i \alpha \\ \delta_2 &= x_i \beta \end{aligned}$$

We pick blinding values and compute the helpers:

$$\begin{aligned} r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} &\xleftarrow{\$} \mathbb{Z}_r^* \\ R_1 &= u^{r_\alpha} \\ R_2 &= v^{r_\beta} \\ R_3 &= e(T_3, g_2)^{r_x} \cdot e(h, \omega)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}} \\ R_4 &= T_1^{r_x} \cdot u^{-r_{\delta_1}} \\ R_5 &= T_2^{r_x} \cdot v^{-r_{\delta_2}} \end{aligned}$$

Using those values, the challenge can be computed:

$$c = Hash(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_r^*.$$

Using c, we now construct the secret holders

$$\begin{aligned} s_\alpha &= r_\alpha + c\alpha \\ s_\beta &= r_\beta + c\beta \\ s_x &= r_x + cx \\ s_{\delta_1} &= r_{\delta_1} + c\delta_1 \\ s_{\delta_2} &= r_{\delta_2} + c\delta_2 \end{aligned}$$

The signature is finally assembled

$$\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}).$$

This signature does not disclose any information about the identity of the signer.

Verification

The verification is performed with the group public key $gpk = (g_1, g_2, h, u, v, \omega)$, the message M that generated the signature and σ, the signature to check.

The verification goes as follow. We first derive the helpers:

$$\begin{aligned}
\tilde{R_1} &= u^{s_\alpha} \cdot T_1^{-c} \\
\tilde{R_2} &= v^{s_\beta} \cdot T_2^{-c} \\
\tilde{R_3} &= e(T_3, g_2)_x^s \cdot e(h, \omega)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \left(\frac{e(T_3, \omega)}{e(g_1, g_2} \right)^c \\
\tilde{R_4} &= T_1^{s_x} \cdot u^{-s_{\delta_1}} \\
\tilde{R_5} &= T_2^{s_x} \cdot v^{-s_{\delta_2}}
\end{aligned}$$

Finally, those recomputed values are used to recreate

$$\tilde{c} = Hash(M, T_1, T_2, T_3, \tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5}).$$

If $c = \tilde{c}$, then the signature is checked and we are sure that it has been signed by a valid group member.

Opening

The opening is the operation that allows to open a signature and identifies the group-member that generated this signature. This algorithm can only be performed with the group manager's key $gmsk = (\xi_1, \xi_2)$. It also needs the signed message M and the signature σ to trace.

The first step is to check the validity of the signature using the previous algorithm. Then we recover the A corresponding to the signature using $T_1, T_2, T_3, \xi_1, \xi_2$

$$A = \frac{T_3}{T_1^{\xi_1} \cdot T_2^{\xi_2}}.$$

This A can now be looked up in the database of the group's secret keys gsk to recover the identity of the user. We assume that A_i, as it is generated by random values within big sets, is unique in the database but theoretically, collisions can happen.

Revocation

Revocation is the action that allows to remove a group member. By revoking him, he will not be able to sign messages anymore. Suppose we want to remove r members of the group. The Revocation Authority (RA) publishes a Revocation List (RL) that contains the revoked user's private keys

$$RL = \left[(A_1^*, x_1), ..., (A_r^*, x_r) \right].$$

Where A_i^* is computed using γ that is the secret group key:

$$A_i^* = g_2^{\frac{1}{\gamma + x_i}} \in \mathbb{G}_2.$$

Note that if $\mathbb{G}_1 = \mathbb{G}_2$, then $A_i^* = A_i$, otherwise $A_i = \varphi(A_i^*)$ and γ is not needed to generate the RL. New keys are generated, based on this RL. The group's public key can be computed as :

$$\begin{aligned}
y &= \prod_{i=1}^{r} (\gamma + x_i) \\
\widehat{g_1} &= g_1^{\frac{1}{y}} \\
\widehat{g_2} &= g_2^{\frac{1}{y}} \\
\widehat{\omega} &= g_2^{\gamma} \\
\widehat{gpk} &= (\widehat{g_1}, \widehat{g_2}, h, u, v, \widehat{\omega})
\end{aligned}$$

Each member also has to update its private key. Those are recomputed as :

$$gsk[i] = (\widehat{A_i}, x_i) = \left(\frac{\varphi(A_1^*)^{\frac{1}{x-x_1}}}{A^{\frac{1}{x-x_1}}}, x_i \right)$$

where (A_1^*, x_1) is the first entry of the RL. This operation has to be executed for every new entry of the RL and the result is securely sent to Provers.

In the revocation mechanism, a user is revoked by the publication of a value that exposes his private key (the entry in the RL). It is thus really important to send the revocation list simultaneously to all Verifiers. Otherwise, someone who obtains a new entry of the revocation list can fool a Verifier who has not yet updated its copy of the Revocation List and keys.

Specificities

This scheme is a true Group Signature Scheme and can thus be used in our use case[5]. The big advantage of this algorithm is the size of the generated signature as it is only 1533 $bits$ = 192 $Bytes$ long when using 170 $bits$ prime numbers and $\mathbb{G}_1$ with members of size 171 $bits$. For the same security with RSA, the signature would have been more than 9 KB long.

The computations required to sign a message are heavy but the verification is light. This is interesting as the signer, in our use case, is a smartphone, that can be really powerful.

However, this scheme poses really strong issues with revocation as every member of the group has to update public keys simultaneously. This scheme needs a good synchronization that is really hard to achieve in practice. Those issues are covered by the Verifier Local Revocation (VLR) scheme.

4.5.4 XSGS (eXtremely Short Group Signature Scheme)

eXtremely Short Group Signature Scheme (XSGS) is a scheme presented by Delerablée and Pointcheval [14] that can produce shorter signatures than in the BBS scheme. This scheme is really close to BBS except that it relies on the eXternal Diffie-Hellman (XDH)[6] and SDH assumptions. The XDH assumption allows to furthermore reduce the size of the signature but it is not crucial for the scheme [14].

Key generation

The key generation produces a group public key gpk, the n group member's secret keys $gsk[i]$, the group revocation key grk and the group manager secret key $gmsk$.

[5]For memory, the use case chosen for this thesis is an access authentication system intended to be used in public events such as concerts or sport events. It is necessary to keep the anonymity of the user (the signer) but we want to be able to revoke this anonymity in case of accident such as a terrorist attack.

[6]Given three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and a pairing function e $\mathbb{G}_1 \times \mathbb{G}_1 \leftarrow \mathbb{G}_T$, while the Decisional Diffie-Hellman (DDH) is easy in $\mathbb{G}_2$, the DDH is said to be hard in $\mathbb{G}_1$ under the XDH [14].

We begin with the *gpk* generation:

$$\begin{aligned} g_2 &\xleftarrow{\$} \mathbb{G}_2 \\ g_1 &= \varphi(g_2) \\ u &\xleftarrow{\$} \mathbb{G}_1 \\ v &= \xi_1 k \\ g &= \xi_2 k \\ \gamma &\xleftarrow{\$} \mathbb{Z}_r^* \\ \omega &= \gamma \cdot g_2 \end{aligned}$$

Finally, $gpk = (g_1, g_2, u, v, g, \omega)$, $gmsk = \gamma$ and $grk = (\xi_1, xi_2)$. We see that this generation algorithm is pretty similar to BBS.

The n group member's secret keys $gsk[i]$ are generated as follow:

$$\begin{aligned} x_i, y_i &\xleftarrow{\$} \mathbb{Z}_r^* \\ A_i &= \frac{g_1 + y_i \cdot h}{x_i + \gamma} \\ gsk[i] &= (A_i, x_i, y_i) \end{aligned}$$

Each $gsk[i]$ can then be transmitted to the appropriate group member.

Signature

As in BBS, we first produces the T_j's values:

$$\begin{aligned} \alpha, \beta &\xleftarrow{\$} \mathbb{Z}_r^* \\ T_1 &= u\alpha \\ T_2 &= A_i + v\alpha \\ T_3 &= \beta u \\ T_4 &= A_i + \beta g \end{aligned}$$

We then pick blinding values and compute the helpers (R_j):

$$\begin{aligned} r_\alpha, r_\beta, r_x, r_z &\xleftarrow{\$} \mathbb{Z}_r^* \\ R_1 &= r_\alpha u \\ R_2 &= e(T_2, g_2)^{r_x} e(v, \omega)^{-r_\alpha} e(v, g_2)^{-r_z} \\ R_3 &= r_\beta u \\ R_4 &= r_\alpha v - r_\beta g \end{aligned}$$

We then compute the challenge which is the hash of the previously computed values and the message:

$$c = Hash(m, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4),$$

this challenge is used to generate the secret values s_j as

$$\begin{aligned} s_\alpha &= r_\alpha + c\alpha \\ s_\beta &= r_b eta + c\beta \\ s_x &= r_x + cx \\ s_z &= r_z + cz. \end{aligned}$$

We finally assemble the signature that can be sent to the Verifier

$$\sigma = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z).$$

Verification

In order to check the signature, the Verifier recomputes the helpers $\tilde{R_j}$'s

$$\begin{aligned} \tilde{R_1} &= s_\alpha u - cT_1 \\ \tilde{R_2} &= e(T_2, g_2)^s_x e(v, \omega)^{-s_\alpha} e(v, g_2)^{-s_z} e(T_2, \omega)^c e(g_1, g_2)^{-1} \\ \tilde{R_3} &= s_\beta u - cT_3 \\ \tilde{R_4} &= s_\alpha v - S_\beta g \end{aligned}$$

Then the Verifier computes the challenge via those $\tilde{R_j}$'s as

$$\tilde{c} = Hash(m, T_1, T_2, T_3, T_4, \tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}),$$

and checks the equality between $\tilde{c}$ and c. If this equality holds, the signature is valid and access is granted.

Opening

The opening is performed by the key manager of the group, given a valid signature. It derives the A used for signing via

$$A = T_2 - \xi_1 T_1.$$

This value can then be searched for in the gsk database in order to retrieve the associated signer's identity.

Revocation

The revocation is fairly complicated and also relies on regenerating new public keys for every group member. This mechanism has thus the same issue as BBS. The group manager generates the following values:

$$\begin{aligned} B_i^* &= \frac{1}{\gamma + x_i} g_2 \\ u_i^* &= \frac{1}{\gamma + x_i} u \\ v_i^* &= \frac{1}{\gamma + x_i} v \end{aligned}$$

Those are sent to every group member (Provers and Verifiers) along with x_i: $(B_i^*, x_i, u_i*, v_i^*)$.

Anyone can now recompute the new public keys:

$$\begin{aligned} \tilde{g_1} &= \varphi(B_i^*) \\ \tilde{g_2} &= B_i^* \\ \tilde{\omega} &= g_2 - x_i \cdot B_i^* \\ \tilde{u} &= u_i^* \\ \tilde{v} &= v_i^* \end{aligned}$$

Each non-revoked signer can also create his new A

$$\tilde{A} = \frac{1}{x - x_i}\left(\varphi(B_i^*) + y \cdot v_i^*\right) - \frac{1}{x - x_i}A.$$

The new $gsk[i]$, computed by the group manager and sent back to the group-members now becomes

$$gsk[i] = (\tilde{A}, x_i, y_i)$$

Specificities

The signature is indeed smaller than in BBS as in this case, those are only 1444 *bits* long (less than 181 *Bytes*). However, this scheme suffers of the same issues as BBS: the revocation is complicated and has to be synchronous for every group member. Indeed, the private key of the revoked user is revealed when revoked. It is thus crucial that every member updates its keys at the same time, especially for the Verifiers.

4.5.5 VLR (Verifier Local Revocation)

As seen in the BBS and XSGS schemes, the revocation is difficult in practice. Indeed, having to synchronize every group-member is really complicated in a mobile and decentralized environment. A solution to this issue is found by Boneh and Shacham: the Verifier Local Revocation scheme [10]. This scheme is a group signature scheme that also uses the SDH assumption and Pairing. It is furthermore pretty similar to BBS.

Key generation

The key generation takes n, the size of the group, as input and generates the group public key gpk and the secret keys $gsk[i]$ for $i \in [0, n-1]$. The group-revocation-token vector $grt[i]$ is also generated. Each entry of $grt[i]$ corresponds to the i^{th} group member and is used to revoke this particular member.

$$\begin{aligned} g_2 &\xleftarrow{\$} \mathbb{G}_2 \\ g_1 &= \varphi(g_2) \\ \gamma &\xleftarrow{\$} \mathbb{Z}_r^* \\ \omega &= g_2^{\gamma} \\ gpk &= (g_1, g_2, \omega) \end{aligned} \tag{4.8}$$

The group-members' keys are tuples $gsk[i] = (A_i, x_i)$ generated using the following algorithm $\forall i \in [0, n-1]$:

$$\begin{aligned} x_i &\xleftarrow{\$} \mathbb{Z}_r^* \backslash \{-\gamma\} \\ A_i &= g_1^{\frac{1}{x_i+\gamma}} \\ gsk[i] &= (A_i, x_i) \\ grt[i] &= A_i \end{aligned}$$

The key manager keeps grt and ω private while he sends the public key gpk to anyone and the $gsk[i]$ to the corresponding user.

Signature

Given a message M to sign, gpk the group public key and $gsk[i]$ the member's key, we can compute the signature.

First, we generate the following values :

$$\begin{aligned} r &\xleftarrow{\$} \mathbb{Z}_r^* \\ (\hat{u}, \hat{v}) &= H_0(gpk, M, r) \in \mathbb{G}_2^2 \\ (u, v) &= (\varphi(\hat{u}), \varphi(\hat{v})) \in \mathbb{G}_1^2 \\ \alpha &\xleftarrow{\$} \mathbb{Z}_r^* \\ T_1 &= u^\alpha \\ T_2 &= A_i \cdot v^\beta \end{aligned}$$

Where $H_0(\cdot)$ is an hash function that maps to $\mathbb{G}_2^2$.

Then, we generate an helper (δ) and we pick blinding values in order to generate helpers:

$$\begin{aligned} \delta &= x_i \alpha \in \mathbb{Z}_r^* \\ r_\alpha, r_x, r_\delta &\xleftarrow{\$} \mathbb{Z}_r^* \\ R_1 &= u^{r_\alpha} \\ R_2 &= e(T_2, g_2)^{r_x} \cdot e(v, \omega)^{-r_\alpha} \cdot e(v, g_2)^{-r_\delta} \\ R_3 &= T_1^{r_x} \cdot u^{-r_\delta} \end{aligned}$$

The challenge can be generated using the previously computed values. Secret holders are also generated, using blinding values:

$$\begin{aligned} c &= Hash(gpk, M, r, T_1, T_2, R_1, R_2, R_3) \in \mathbb{Z}_r^* \\ s_\alpha &= r_\alpha + c\alpha \in \mathbb{Z}_r^* \\ s_x &= r_x + c x_i \in \mathbb{Z}_r^* \\ s_\delta &= r_\delta + c\delta \in \mathbb{Z}_r^* \end{aligned}$$

Finally the signature is

$$\sigma = (r, T_1, T_2, c, s_\alpha, s_x, s_\delta).$$

Verification

The signature verification is composed of two operations. The first one is the validity check and the second the revocation check. The validity check is performed by the following algorithm :

$$\begin{aligned}
(u, v) &= (\varphi(\hat{u}), \varphi(\hat{v})) \in \mathbb{G}_1^2 \\
\tilde{R_1} &= \frac{u^{s_\alpha}}{T_1^c} \\
\tilde{R_2} &= e(T_2, g_2)^{s_x} \cdot e(v, \omega)^{-s_\alpha} \cdot e(v, g_2)^{-s_\delta} \cdot \left(\frac{e(T_2, \omega)}{e(g1, g2)} \right)^c \\
\tilde{R_3} &= T_1^{s_x} u^{-s_\delta}
\end{aligned}$$

Then it regenerates $\tilde{c} = Hash(gpk, M, r, T_1, T_2, \tilde{R_1}, \tilde{R_2}, \tilde{R_3})$ and compares c and $\tilde{c}$, if those matches, then the signature is valid, otherwise it is rejected.

In case of a valid signature, the Verifier now has to check that the signer is not revoked and is still authorized to generate signatures. This verification consists of an iteration on a RL. The RL contains the A_i of the revoked user. The Verifier computes the equality

$$e\left(\frac{T_2}{A_j}, \hat{u}\right) = e\left(T_1, \hat{v}\right)$$

for each user $A_j \in RL$ if this equality holds, the member corresponding to A_j is revoked.

As we can see, this verification complexity grows linearly with the size of the RL. This can be an issue and is discussed in Chapter 5.

Revocation

The revocation of a user is performed by the use of grt. By adding $grt[i]$ to the RL, group member i will be revoked. The RL is thus a simple list of elements from grt. This list has to be sent to the Verifiers. The other group members do not need access to the RL which solves the main issue of BBS and XSGS schemes.

Opening

The opening in VLR is exactly the same principle as the revocation check. Only the key issuer (which has access to grt) can open the identity of a signature.

The idea is to first check the signature. If it is valid and non-revoked, we can start the following algorithm:

1. Create a temporary list of plausible identities from grt.
2. Cut this list in two equal parts
3. Check the signature against one of the two previous lists.
4. If the signature is revoked with this list, then the identity is in the tested list. If it is not revoked, then the identity is in the other list.
5. Restart at step 2 with the revoked list.

The opening is thus a binary search amongst a list of plausible identities. The computational complexity follows $\mathcal{O}\left(log_2(n)\right)$ where n is the size of the group. This complexity is not really important as the opening is performed offline, having access to grt and the signature to open.

Specificities

This scheme has some interesting specificities. The most important one is the ability to revoke a user without disclosing any information to the entire group. This means that the revocation is a lot simpler than in BBS. This also means that group members can be completely disconnected. This has a big importance in systems involving a lot of parties like group signature as it is really difficult to keep every parties up to date. Furthermore, as no private information is disclosed when the user is revoked, the synchronisation of every Verifier is not as critic as in BBS.

The key-size is also smaller for the same security (elements of $\mathbb{G}_1$ are size 171 *bits*): The total signature-size is 149 *Bytes*.

In term of computational complexity, the verification step is the most problematic. As its complexity increases linearly with the size of the Revocation List, a trade-off has to be found. It is still possible to regenerate new keys when the RL becomes to long. Those new keys will have to be transmitted to the non-revoked users and synchronized really carefully as the previously issued keys becomes invalid. Technical solutions for this kind of mechanism and definition of this critical point are discussed in Chapter 5.

4.6 Summary of analysed schemes

This section briefly summarizes the functional differences between schemes presented before. First of all, BB and BLS are different as those are not group signature schemes. They allow to sign messages asymmetrically but not anonymously. We can hence ignore them for the choice of the scheme.

BBS, XSGS and VLR are group signature schemes that allow to sign messages anonymously. Each scheme provide revocation and open mechanisms which are really important for the use case studied by this Thesis. However, VLR presents a more practical revocation as it does not need a perfect synchronisation between each group-member. We will thus consider VLR as more easy to use in real systems.

4.7 Conclusion

This chapter detailed some mathematical constructions that allow group signatures. We reviewed the basis of ECC and PBC in order to understand the operations performed by schemes such as BB, BLS, BBS, XSGS and VLR. Those schemes have been discussed in their mathematical aspects and their specificities for our use case that is, for memory, the use of token authentication in public events such as concerts and festivals. The chosen scheme has to protect the identity of the user as long as no problem occurs. In case of an unlikely event such as a terrorist attack etc., it must be possible to open the signature and see the actual identity of the issuer.

The revocation issue has also been discussed as it is really important in such system to revoke an access in case of fraud, default of payment, etc. We saw that Verifier-Local-Revocation (VLR) scheme fits those needs and is the most comfortable to work with. We will now discuss its implementation in Chapter 5.

Chapter 5

Implementation: Proof of Concept

5.1 Introduction

This chapter focusses on the implementation of a POC for the chosen use case. For memory, the aim of the system is to be used in public events and transportation. The system has to provide privacy for the user but also the ability to revoke a user. Also, in case of an unlikely event such as a terrorist attack, it must be possible to identify people involved in it and therefore revoke the anonymity. This system has to be portable, low cost and as easy as possible to use for the user.

The system will rely on PBC and schemes described in Chapter 4. We will discuss those schemes and implement one. The implementation is tested against benchmarks which results are presented. This chapter is more practical as it explains every implementation choices made.

5.2 Use case details

The designed system is aimed to be used in public events such as concerts. Let us imagine a subscription paid on a regular basis that allows the user to attend any concert of a city. Every paying user has access to every concert if the subscription is paid. In such a system, the only information needed at the verification (when the user wants to join a concert) is the state of the subscription (revoked or not). The identity of the user is not important at that stage and furthermore, the user can be reluctant about giving his identity at the entrance.

5.2.1 Group-size

In such system, the group size has to be fixed. Let us take the case of Brussels. The city has multiple concert halls and provides a lot of concerts every week. We can imagine a group size of 100000 *people*. With such an amount, we can reserve room for new subscribers and to give access back to revoked ones if needed. This group-size is also interesting as it is large enough to imply long enough computations.

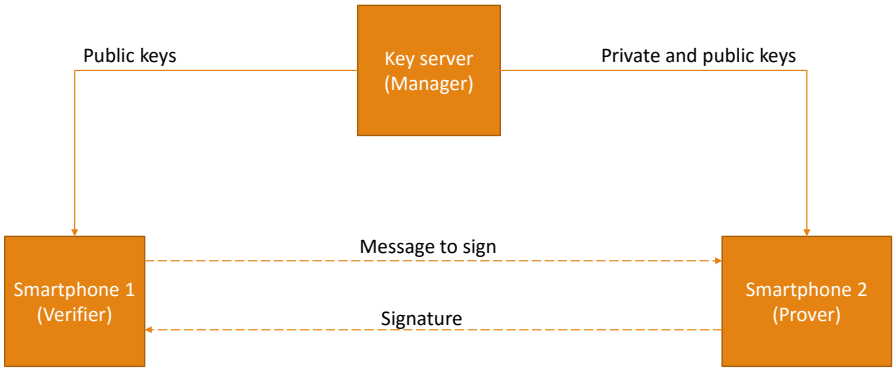

Figure 5.1: System architecture of the POC

5.2.2 Computational power

As we need to use active tags for PBC, we will use smartphones. Indeed, nearly everyone has a smartphone these days and those can provide enough computational power to sign messages. NFC is present on a lot of smartphones which can allow using access authentication systems as those described in Chapter 1.

In the case of smartphones, those can be connected to the web but the communication cannot be trusted and this connection is not permanent. Meaning that we cannot rely on it for instantaneous messages. The bandwidth used also has to be minimal in order to reduce costs as less as possible for the user.

5.3 System architecture

The system is composed, for simplicity of three entities:

Key manager : It generates the keys for the whole group and is in charge of revoking/opening the identities behind signatures. Its power can be considered "unlimited" as it can delegate the key generation to cloud instances. This aspect is discussed in the benchmark section (Section 5.5).

Verifier : The Verifier sends a message to the Prover and checks its signature. The Verifier is a smartphone (for simplicity of implementation[1] in our case). It can also be an Access controller that can handle NFC such as the SC415 with the STid ARC-A as reader.

Prover : The Prover signs received messages from the Verifier. The Prover is the user's smartphone that runs a specific Android application for the system.

In a real system, it can be interesting to use four entities and to delegate the revocation and opening to another party.

Figure 5.1 depicts the system as it is implemented for this POC. The key manager is a simple Linux computer and the two other entities (Verifier and Prover) are two Android smartphones[2].

[1]The architectures are very different between an Android smartphone and the SC415. The choice has been made to only implement one application in order to prove the concept.

[2]For the demonstration, a LG Nexus 5 and an Huawei P8 Lite are used. Those two are NFC compatible and are ARM based.

Communication between the Key server and smartphones is done via a secured socket connection and the communication between smartphones is performed via NFC. The Prover acts as a tag via the ISO 21481 [22] specification.

5.3.1 Simplification for the POC

The system has been simplified for this POC as the NFC communication was not working[3]. Every communication is now performed via a socket connection.

For this thesis, the socket connections are supposed secure, the secure implementation of those connections is out of the scope of this thesis but it is straightforward to imagine a secured connection over SSL or TLS for example. In fact, the communication protocol is really simple and reduced to the most basic implementation possible. This protocol is also detailed in Section 5.6.

5.4 Libraries and technical choices

This section summarizes the libraries used for the implementation but also the technical choices made in order to construct a working POC that can demonstrate the usability of such a system.

5.4.1 Crypto libraries

libPBC The cryptographic part relies on libPBC [25]. This library, written in C by Benn Lynn, implements pairing functions. It is open-source and licensed under LGPL (see Appendix **??** for the full licence). Its particularity is to be abstracted such a way that it allows the user to only have an elementary understanding of the pairing. Theory exposed in Chapter 4 is sufficient to understand the use of the library. An example of libPBC usage is presented in Appendix **??**.

libGMP The libPBC library relies on libGMP[4] which is a free library for arbitrary precision arithmetic. It allows to use big numbers such as those used in cryptography. GMP is embedded in several Linux distributions and is also licensed under LGPL.

OpenSSL Hash functions used for this implementation are using OpenSSL[5], available under Apache licence [6]. This library, widely available on Linux systems, embeds hash functions such as SHA1, SHA256,... This POC uses SHA256 implemented in OpenSSL.

5.4.2 Mobile libraries

Android SDK/NDK At the beginning, the Android Software Development Kit (SDK)[7] was chosen. This Java SDK is used by native applications and can embeds C libraries such as libPBC and OpenSSL. This integration is done via the Android NDK[8]. Unfortunately, the documentation for this kind of integration is not easy to find and understand. The integration using JNI[9] is really complicated and the Android Compiler adds a lot of difficulties as the libraries have to

[3]This issue is explained further in detail in Section 5.6.

[4]More information about libGMP can be found on the official website `https://gmplib.org/`

[5]The official OpenSSL website is `https://www.openssl.org/`

[6]The full licence is available at `https://www.openssl.org/source/license.html`.

[7]The official Android SDK and documentation website is `https://developer.android.com/guide/`.

[8]The official website and documentation of the NDK is `https://developer.android.com/ndk/`

[9]JNI stands for Java Native Interface, its documentation can be found at `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/jniTOC.html`.

follow strict rules in order to be compatible with Android devices. This solution was abandoned after some weeks of struggling with the different architectures and compatibility issues.

Qt Qt is another open-source SDK that allows to create mobile applications. The licence is also LGPL license that can be found in Appendix **??**. Qt is a C++ library that allows to interact with the Android SDK via modules such as QNFC, QNetwork,... Those handle the links and the data-type conversions.

Furthermore, embedding a C library in a C++ program is a lot more easier and documented than JNI. The choice for Qt was also motivated by the fact that it is multi-platform. This means that the implementation is compatible with computers such as Windows, Mac OS X and Linux but also with mobile devices. The implementation particularly focusses on Android smartphones but the project can easily be ported to iOS (note however that NFC is currently not available on iPhone), Windows 10 and even Blackberry 10 phones.

5.5 Benchmarks

The benchmarks are performed on an Intel Core i5-2430M CPU at frequency 2.40 GHz under Linux (Ubuntu 15.04). Those Benchmarks enabled to measure performances of the different schemes.

5.5.1 Schemes comparison

First, it is interesting to benchmark the different schemes in order to compare them and check the feasibility of the system. The following schemes have been implemented and benchmarked: BLS, BB, BBS and VLR. XSGS was not implemented as it is pretty similar to BBS in term of complexity. In addition, the revocation and opening mechanisms are the same in BBS and XSGS.

Every benchmark is executed 10 times in order to better estimate the time and reduce the estimation bias. The benchmarks are first performed with 1000 keys, then 10 000 and finally 100 000 keys in order to understand the impact of the group-size on the different operations. We expect the key generation time to be linear with the group-size as we generate them in a loop.

Figure 5.2 represents different boxplots[10] showing the time spent for each scheme in the main operations. The initialization is performed once per system installation. It generates cyclic groups and randomness for next operations. We can see that the generation time is close for every scheme This generation is pretty similar for every scheme. The main difference is in the key generation where BB is really slower than the other schemes. Indeed, BB is the only scheme that requires pairing in the key generation phase. As pairing is a lot more complicated than rational operations and multiplication of group elements, this difference is expected.

VLR and BLS are slower for signature and verification which are the online operations. This difference was also expected as the signature and verifications are very complex compared to BLS and BB ones (see Chapter 4). In general, those operations have to be performed in less than $500ms$ in order to be imperceptible for the user. We can see that this condition is met as the median time is equal to 35 ms for BBS and 61 ms for VLR.

By increasing group-size, the time needed to generate keys increases linearly with the group-size. We can easily imagine bigger groups, as the key generation can be made by cloud clusters, this step is not critic. We can see this relation on Figure 5.3. We can also notice that the

[10]The red line is the median time, the blue box represents the 25 and 75 percentile, the whiskers extends to the most extreme point that is not an outlier and the red crosses show the outliers.

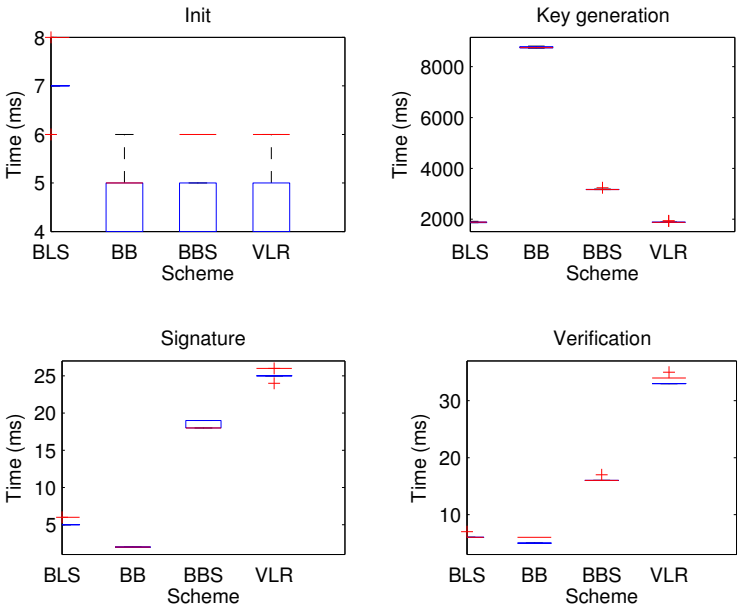

Figure 5.2: Time comparison for 1000 keys (boxplots[10])

signature and verification time does not change when increasing the group-size. In fact, only the RL increases the verification step. We can expect a linear complexity with the RL's size.

We will now focus on VLR as this scheme is the only one that can address the issue of the revocation. Indeed, it is the only scheme that enable to revoke users without having to regenerate new public and private keys. We also pointed out its feasibility with respect to the timing and computational constraints.

5.5.2 VLR: Curves comparison

It is interesting to compare different types of curves. Those have different security levels and performance. Previous benchmarks where performed using A curves. Those have a security level equivalent to 1024 *bits* keys in Discrete log cryptography but their base field's size is 512 *bits* which is pretty big compared to other curves.

Table 5.1 summarizes different types of curves tested. This table also displays the time needed for every operation of VLR scheme (initialization, key generation for 10 000 keys[11], signature and verification with 100 revoked users). Timing is also represented visually in Figure 5.4.

Curves of A type are really faster than the others, except at key generation where $D159$ and $D201$ keys are almost twice faster. Furthermore A curves take too much space for our system. Indeed, those are two to three time bigger than the others analysed curves. As initialization and key generation times are not critical, only the signature and verification are important. Those two must ideally stay below 500 ms for the system to stay perceived as real-time for the final user.

[11] We used 10 000 keys in order to reduce the key-generation phase. As we already saw the linearity with the group-size, we can take the hypothesis that this linearity still holds with different classes of curves.

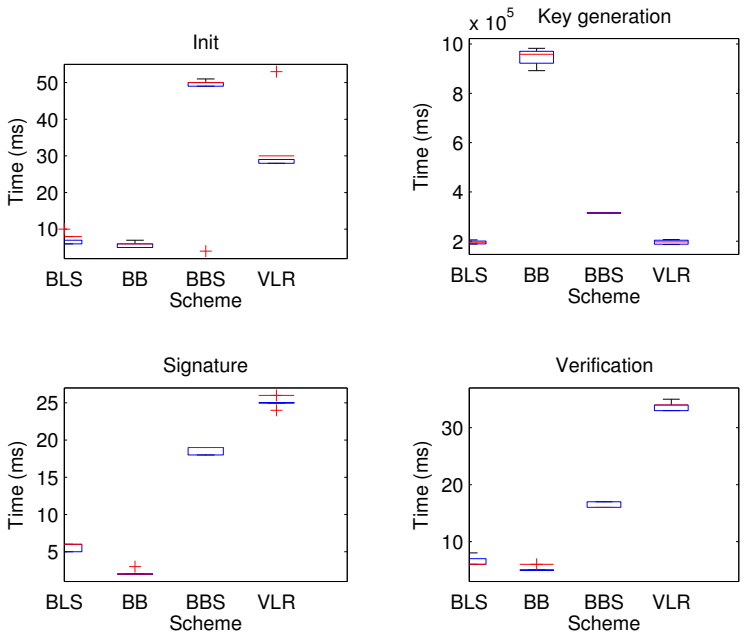

Figure 5.3: Time comparison for 100 000 keys (boxplots[10])

Looking at Table 5.1, we see that using $D159$ or $D201$ curves is a nice trade-off between size and performances. Also, the security of those curves is slightly better than with A curves. As a result of this comparison, $D201$ curves are used for the rest of the Thesis.

5.5.3 VLR: Revocation List sizes benchmarking

As verification time grows linearly with the RL's length, it is also interesting to compare different sizes of RL in order to assess the limit. Indeed, at some point, it will be necessary to regenerate keys for the whole group in order to get a new empty RL. This critical point has to be found.

We try to keep the processing time below 500 ms. By keeping delays below this limit, we ensure that users will be tolerant to the waiting. The highest limit for usability is around 2 s. Delays longer than 2 s are too long and impact the usability of the system. Figure 5.5 shows the evolution of the verification time with the Revocation List length. Green and Red lines are the

	A	**D159**	**D201**	**F**
Field- base size (bits)	512	159	201	160
Degree of the polynom	2	6	6	12
Equivalent security in Dlog (bits)	1024	954	1206	1920
Initialization time (ms)	5.1	27.3	46.3	29.28
Key generation (ms)	1893.4	834.6	1102.1	8470
Signature (ms)	24.6	19.2	28.301	89.4
Verification (ms)	159.1	452.6	654.9	2515.8

Table 5.1: Curves comparison (VLR)

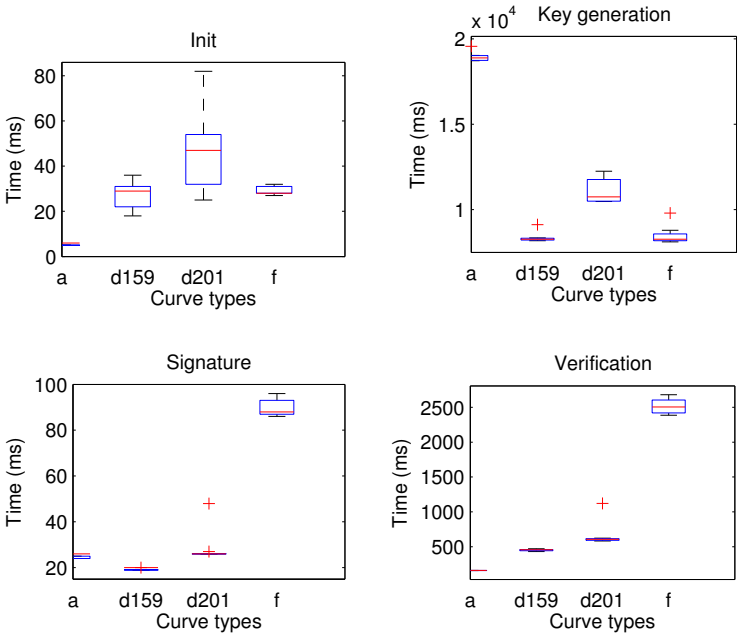

Figure 5.4: Time comparison for different Curve types (boxplots[10])

limits of 500 ms and 2 s in this figure.

We can see that a very small amount of revoked users causes the verification delay to become too long. Indeed, only 80 revoked users correspond to a verification time of 500 ms. For real-time systems, a group reset is thus necessary in order to keep this verification time low and a comfortable system. We can accept a delay up to two seconds, the hard limit for the size of the Revocation List is hence 350 revoked users.

VLR also suffers of the same issue than BBS as it is also necessary to regenerate new keys at some point. The difference is that in BBS, this regeneration is needed after one revocation. With VLR, we can delay the operation and thus simplify the system. Also, the revoked private keys are not disclosed publicly. However, this means that for each reset of the group, each user has to receive, via secured channel, its new secret key. This can lead to complicated systems and this thesis does not cover them.

5.6 Implementation of the Proof of Concept

The Proof of Concept has been implemented on Android for the Clients (Verifier and Prover) and on Linux for the Server (Key manager). Those implementations are based on the Qt library. Source code is available on request and is also published on Github[12].

5.6.1 Software architecture

The Proof of Concept follows the Model View Controller (MVC) paradigm: Software is cut into three parts in order to ease development and code maintainability.

[12]See `https://github.com/Gp2mv3/ThesisPOC`

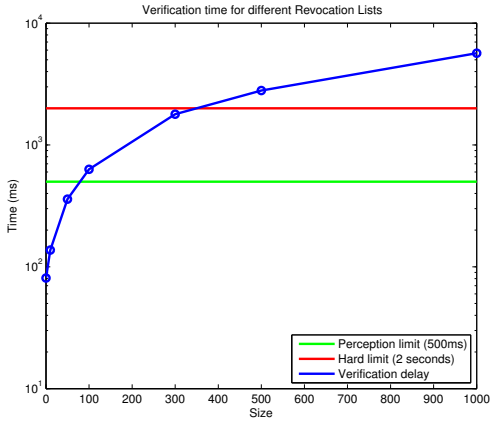

Figure 5.5: Validation time versus different sizes of Revocation List (boxplots[10], log scale)

Models : Contains the data that the software handles but also data sources. There are multiple models such as keys (private, public and revocation tokens), signatures, messages and Revocation List. The Network is also assimilated as a Model as it provides new data to convert.

Views : Views define how to display Models. In this application, views are simple and only one view form is used. This form defines the graphical aspects of the entire application.

Controllers : Controller handles the logic of the application. In our case, it orchestrates Network, Views and cryptographic operations.

The application embeds two wrappers: one for network and communications and the second for cryptographic operations. The main controller handles links between those two elements and the logic of the application.

Qt defines a system of signals and slots to handle asynchronous events. Those can be user interactions but also events from Network, computational operations, memory etc. In our case, cryptographic operations are synchronous and do not use signals and slots, for simplicity.

5.6.2 Client-side

The application is the same for every parties of the scheme: Verifier, Prover and Server run the same application. This choice is more convenient in case of code modification but also for a faster development.

Figure 5.6 shows screenshots of the Android application. This application contains four tabs, the first one is dedicated to network configuration. It allows to enter IP addresses of the Server and a Verifier.

The second tab is dedicated to authentication. The screen contains a button to request fresh keys from the server. This button simply sends a request to the server that responds with a new non-revoked user secret key. The second button is used on the Prover. It asks the Verifier a message to sign. The Verifier answers with a new random message. This message is automatically signed by the Prover and its signature is transmitted to the Verifier. At reception of a signature, the Verifier checks its validity with the last message sent and a pop-up signals the result of the verification (Valid, Invalid, Revoked).

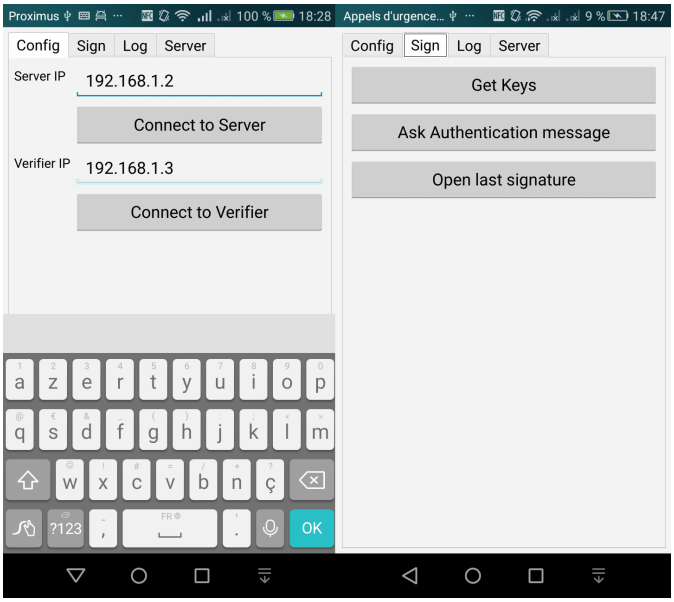

Figure 5.6: Screenshots of the Android client application

Finally the last tabs are simply a log screen that logs every operation and the server tab.

5.6.3 Server-side

The server is also based on the same Qt application for a simpler communication with the client application. In fact the fourth tab of the application is dedicated to the server and it is only used for group keys generation and key revocation. The server generates keys (100 000) and keeps them in memory. To every new request for a new key, the server responds with an unused private key and the group public key.

Figure 5.7 shows the server window and the list of currently distributed keys. The server can revoke a key from the list (each row is clickable). When a key is revoked, the revocation message is sent to every connected devices.

5.6.4 Communication

For simplicity, socket connections are used between group-members. Every request and answer transit on the network via Transmission Control Protocol (TCP) on port 8081. A request always generates an answer, even if it is empty. This convention can be used as an acknowledgement.

Message types are listed in Table 5.2. Uppercase identifiers designates requests where lower case message corresponds to the answer to the corresponding request.

Serialization is performed in order to send data-structures on the network. Each element (converted to byte array), is preceded by its size on two Bytes. Each packet sent also begins with its total size in order to handle fragmentation that can happen on the network.

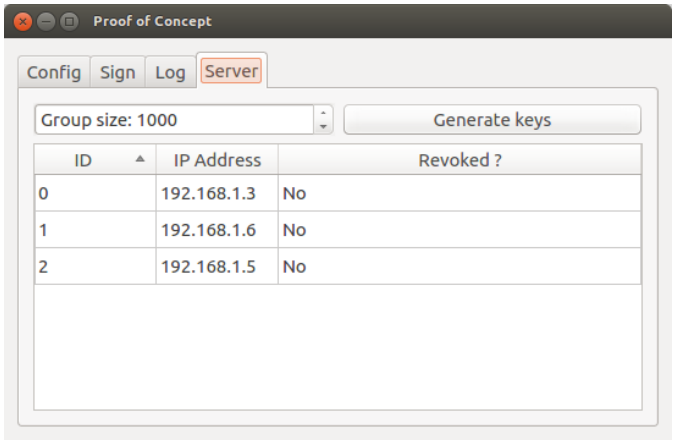

Figure 5.7: Screenshot of the Server application

Identifier	**From**	**To**	**Payload**	**When is it used**
G (Get keys)	V or P	S	/	Request new key
g	S	V or P	Keys (gpk, gsk)	Answer to new key request
A (Auth.)	P	V	/	Prover asks message to sign
a	V	P	Message	New message is generated
S (Sign)	P	V	Signature	New message to sign received
s	V	P	State (Valid, Invalid, Revoked)	Signature is checked
R (Revoke)	S	V or P	Revoked Identity	A key is revoked by the server
r	V or P	S	/	Revocation message is received
O (Open)	V	S	Signature and message	Opening is requested
o	S	V	Identity	Identity is open

Table 5.2: Communication messages of the system (V = Verifier, P = Prover, S = Server)

5.7 Weaknesses and improvements

Computational optimisations are possible in order to reduce the time complexity of operations for the whole system. Indeed, some pairings can be pre-computed and published along with the keys. Those optimisations can reduce the time needed for signature but not the verification time which is the critical one. Indeed, this operation is really straightforward and no optimisation is possible as the pairing is done for every revoked user. It is impossible to pre-compute or cache such a loop.

The client-application is the same for every parties of this scheme. This choice was made for simplicity but in a real system, each parties must have dedicated applications such as a verifier cannot prove any signature and vice-versa. Currently a Verifier also receives secret keys as it is treated equally to Prover. It is also possible for a Prover to act as Verifier if its IP address is configured as Verifier by another Prover. This application is still a Proof of Concept and cannot be used as is.

Currently, every communication is performed through socket connections. Again, the choice was made to use socket connections to reduce the complexity of implementation as NFC on Android is currently not well supported by Qt. Also, a portage of the Verifier on the Access Controller can be interesting but complex as the current application relies on Qt which is not fully compatible with the Access Controller. Furthermore, the architecture and embedded libraries are not the same on Android and the embedded Linux of the SC415.

5.8 Conclusion

This chapter explained the implementation of a group signature scheme for a real system. The use case for this Proof of Concept was a subscription to concert hall with up to 100 000 members that can attend concerts. It is even possible to generate bigger groups without impacting signature and verification operations. The system relies on Smartphones that run a specific application in order to authenticate.

The implemented system relies on TCP connections although NFC is more interesting in term of security and power consumption. This choice was made for simplicity of implementation and because some problems occurred with QNFC (the NFC library of Qt) on Android.

The cryptographic scheme chosen is VLR because it allows to revoke users without having to disclose secret information. It is thus possible to create a scheme that does not need perfect synchronisation between each parties. However, verification time grows linearly with the size of the Revocation List, meaning that a critical point exists. This critical size is relatively small. It is therefore necessary to reset the group and re-generate new keys for every valid member when this critical size is reached.

Different schemes and curves types have been benchmarked in order to obtain a good trade-off between computational complexity, security and timing. We saw that $D201$ curves enable to have a great security level with good performances compared to F or A curves.

Finally, the Proof of Concept, running on Android is explained. This Proof of concept implements VLR scheme with $D201$ curves. This implementation is done with Qt and runs on Android smartphones. Communications are transmitted through socket connections. A key-management server is able to generate group keys and to revoke users if needed.

Conclusion

This thesis covered access authentication with a critical regard on privacy issues. We first focused on basis of access tokens and security of current RFID systems. Those systems present flaws in term of security and privacy that can be patched by improving them.

As an example, Chapter 2 explained a MIFARE DESFire implementation and how it works in details. Similar systems are used in access authentication, public transportations and even ticketing although it does not protect the user's privacy. Also, those systems have problems related to key storage as keys are often stored in the reader which is located outside the secured area. Improvements have to be applied such as Transparent mode which allows to use the reader as a simple relay with no critical information stored in it. This improvement is detailed in Chapter 3 and its implementation is explained. This chapter covers modifications made to a classic system in order to use transparent mode.

With Transparent mode, we cover some security issues presented by access authentication systems but privacy issues remain present. Those issues are important as authentication systems have a lot of use cases such as ticketing and public transportations where the identity of the user can remain hidden. Indeed, in those case, only the authentication of access is important and not the identification. In such cases, group signatures can be exploited in order to ensure privacy for the users. Different schemes exist and we focussed on pairing based schemes such as BBS, XSGS and VLR as those schemes enable revocation and opening of signatures. We discussed those schemes theoretically in Chapter 4. Those where also implemented and benchmarked in order to compare performances and timing. Those benchmarks also allowed to verify the feasibility of a proof of concept. Based on those analyses, Verifier Local Revocation scheme was selected and implemented on Android smartphones. We pointed out that VLR allows revocation without directly regenerate the whole key set. However, we saw that verification times grows linearly with the amount of revoked users. At some point, it is important, in order to keep the revocation time under two seconds, to regenerate the whole group. This critical point is reached at only 300 revoked users which means that VLR also suffers of similar issues as BLS or XSGS. Those issues are hence only delayed. Indeed, in BLS and XSGS, group keys are regenerated at each revocation. Chapter 5 describes those benchmarks and implementations. A proof of concept, developed for this Thesis is available on Android and helps to understand how such systems can be implemented.

As a conclusion, we can say that current authentication systems, used in many situations, still present flaws, those can be patched in order to reduce their scope and risks. However, we also pointed out some use cases which implies new considerations such as privacy preservation. Systems have to be adapted specifically to those usages and have to implement different schemes. Group signature is a solution to privacy issues and we proved its feasibility with a proof of concept.

Bibliography

[1] Priya Agrawal, Neha Bhargava, Chaitra Chandrasekhar, Al Dahya, and J.D. Zamfirescu. The MIT ID card system: Analysis and recommendations. *MIT6.805*, 2004.

[2] Smart Card Alliance. *Host Card Emulation (HCE) 101.* Mobile & NFC Council, 191 Clarksville Rd. Princeton Junction, NJ 08550, aug 2014.

[3] ANSSI. *Sécurité des technologies sans-contact pour le controle des accès physiques.* ANSSI: Agence Nationale de Sécurité des Systèmes d'Information, 1.0 edition, nov 2012.

[4] ANSSI. Rapport de certification ANSSI-CSPN-2013/08: Lecteur RFID LXS w33-e/ph5-7ad, oct 2013.

[5] Gildas Avoine. Access tokens. In *INGI2144: Secured Systems Engineering*, 2015-2016.

[6] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *Advances in Cryptology – EUROCRYPT '03*, 2003.

[7] Alexandra Burlacu. Samsung galaxy note 6 could pack 10nm 6 GB LPDDR4 DRAM. `http://www.techtimes.com/articles/159903/20160520/samsung-galaxy-note-6-could-pack-10nm-6-gb-lpddr4-dram.htm`, may 2016.

[8] D. Chaum and E. van Heyst. Group signatures. *EUROCRYPT'91*, 1991.

[9] Craig Costello. *Pairings for beginners*, nov 2012.

[10] Hovav Shacham Dan Boneh. Group signatures with verifier-local revocation. *11'th ACM conference on Computer and Communications Security (CCS)*, 2004.

[11] Hovav Shacham Dan Boneh, Ben Lynn. Short signatures from the weil pairing. *Journal of Cryptology*, 2004.

[12] Hovav Shacham Dan Boneh, Xavier Boyen. Short group signatures. *Advances in Cryptology—CRYPTO 2004*, 2004.

[13] Xavier Boyen Dan Boneh. Short signatures without random oracles and the sdh assumption in bilinear groups. *Eurocrypt 2004*, 2014.

[14] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. *VIETCRYPT*, 2006.

[15] Morris Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication, may 2005.

[16] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. Wiley, 2010.

[17] Xavier Torrent Gorjon. Protecting against relay attacks forging increased distance reports. Master's thesis, Universiteit van Amsterdam, 2015.

[18] G.P. Hancke, K.E. Mayes, and K. Markantonakis. Confidence in smart token proximity: Relay attacks revisited. *Computers & Security*, 28, oct 2009.

[19] ISO. Identification cards – contactless integrated circuit cards – proximity cards. Standard ISO 14443:2008, International Organization for Standardization, Geneva, CH, 2008.

[20] ISO. Information technology – radio frequency identification for item management. Standard ISO 18000:2008, International Organization for Standardization, Geneva, CH, 2008.

[21] ISO. Information technology – security techniques – entity authentication. Standard ISO 9798:2010, International Organization for Standardization, Geneva, CH, 2010.

[22] ISO. Information technology – telecommunications and information exchange between systems – near field communication interface and protocol -2 (nfcip-2). Standard ISO 21481:2012, International Organization for Standardization, Geneva, CH, 2012.

[23] Marc Joye, Atsuko Miyaji, and Akira Otsuka. *Pairing-Based Cryptography - Pairing 2010*. Springer, 2010.

[24] F. Koeune and O. Pereira. Zero knowledge proofs. In *LMAT2450: Introduction to Cryptography*, 2014-2015.

[25] Ben Lynn. PBC library. `https://crypto.stanford.edu/pbc/`, 2006. LGPL.

[26] Kerry Maletsky. *RSA vs ECC Comparison for Embedded Systems*. Atmel, jul 2015.

[27] Alfred Menezes. An introduction to pairing-based cryptography. *University of Waterloo*, 2005.

[28] NCS Scaline, Paepsem Business Park Bvld Paepsemlaan 18C, 1070 Brussels. *SC4X5 Manual*, apr 2016.

[29] Michael A. Nielse and Isaac L. Chuang. *Quantum computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[30] NXP. *UM0701-02: PN532 User Manual*. NXP Semiconductors, nov 2007. Confidential.

[31] NXP. *AN0945: MIFARE DESFire EV1 - Features and Hints*. NXP Semiconductors, jun 2011. Confidential.

[32] NXP. *MF3ICD81: MIFARE DESFire EV1*. NXP Semiconductors, feb 2011. Confidential.

[33] NXP. *PN532/C1: Near Field Communication (NFC) controller*. NXP Semiconductors, sep 2012.

[34] NXP. *AN11704: MIFARE DESFire Application Guidelines*. NXP Semiconductors, jun 2015. Confidential.

[35] NXP. *K61P144M120SF3*. NXP Semiconductors, oct 2015. Confidential.

[36] NXP. *MF3ICD(H)Q1: MIFARE DESFire EV1 256B contactless smartcard IC*. NXP Semiconductors, nov 2015.

[37] Hovav Shacham. *New paradigms in signature schemes*. PhD thesis, Stanford University, 2006.

[38] STid. *Communication Protocol: SSCP V1*. STid Electronic Identification, MIFARE global edition, sep 2013. Confidential.

[39] STid. *Communication Protocol: SSCP V2*. STid Electronic Identification, MIFARE global edition, sep 2014. Confidential.

[40] Greg Zaverucha Tony Rosati. Elliptic curve certificates and signatures for: NFC signature records. *Research In Motion, Certicom Research*, jun 2011.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**