Universitat
Autònoma
de Barcelona

escola
d'enginyeria

5211 - SECURITY IN RFID DEVICES

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Marti Berini Sarrias i dirigit per Jordi Herrera Joancomartí.

Bellaterra, juny de 2013

El firmant, Jordi Herrera Joancomartí, professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Marti Berini Sarrias

Bellaterra, juny de 2013

Firmat: Jordi Herrera Joancomartí

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we will introduce the concept of RFID and the contents of this project.

RFID (Radio Frequency Identification) technology is the use of a wireless non-contact system to transfer data between a tag and a reader, for the purposes of automatic identification, as a substitute inter alia of the barcode [9].

In the last years that kind of systems have become very popular and a research area of priority [4]. These systems are used in areas like industry, logistics, distribution, access control, e-passports and ticketing.

1.1 Motivation

Let's look a little deeper into the applications of RFID systems: as we can see in Figure 1.1 there are multiple ways to use RFID technology for payment and authentication but the main benefits can be the main disadvantages as well if security is not well implemented. It is contactless, so information can be intercepted from some distance. It is fast, so information can be retrieved or altered almost instantly. Also, it is widely used as an access control system (as in the metro of London) and as a wallet to store credit for some services (as in some vending machines).

All these kinds of applications and many others involve a strong security factor which must be taken into account.

1.2 Objectives

In recent years the use of RFID devices has increased in various services, payment, access control and labelling. An example of these devices are called contactless cards, either in

(a) Public transport (b) Credit cards (c) Event ticketing

Figure 1.1: Applications of RFID

the form of credit cards, Bicing card (Barcelona's bicycle sharing system), activation of alarms or access cards to gyms and hotels. Most systems based on smartcards on the market that use MIFARE family of chips, specifically the Classic model. MIFARE products are technologies that claim to follow the ISO / IEC 14443, but its implementation and protocol is proprietary.

1.2.1 Main objective

Since the implementation of the MIFARE Classic smartcard is secret it's reliability has not been officially verified by others than its creators. It's a risk to rely on a technology that bases its security on obfuscation, but currently it is used in a wide range of services.

It would therefore be interesting to analyse the security of this RFID devices. Since they work by proximity and without direct intervention from the owner, it's important to see to what extent they are reliable, check if they have cracks or security vulnerabilities and study what should be done to correct these problems.

That's why we have decided that the overall objective of the project is to check the security of MIFARE Classic smartcards and the systems which are usedin these devices, analysing its functioning and looking at its potential vulnerabilities.

1.2.2 Specific objectives

Below, we thought it would be interesting to include a list for a better definition of the specific objectives of the project and its sections, and for a better tracking of the work done.

1. Analyze the structure and general operation of RFID infrastructure and how they benefit or harm systems that use them.

2. Study the performance of the communication protocols and security systems that implement the MIFARE Classic tags

3. Study existing known vulnerabilities in systems based on MIFARE Classic chips and see if we can find new ones.

4. Learn to use the specific hardware for reading, writing and sniffing RFID tags.

5. Check the feasibility of the theoretical known vulnerabilities in real world scenarios.

6. Analyze real case security systems implementing MIFARE Classic technology and see the use made of it.

7. See how much the smartphone's NFC is a potential risk factor for RFID systems, if you can replace the specific hardware and operating systems or if they implement some kind of security or control.

8. Search strategies and best practices to maximize the security of RFID systems or at least minimize the risk.

1.3 State of the Art

While NXP kept in secret the implementation of the MIFARE chips, there have been cases of attacks to this technology. As seen in the Chaos Communications Congress [18] the hardware can be reverse engineered regardless the size of chips, so security by obfuscation is not effective in this area. The following year Peter Van Rossum [11] published how they reverse engineered the protocol and the algorithm, along with some practical Attack to MIFARE Classic chips. Later Henryk Plöetz and other people uploaded to the internet detailed information from MIFARE Classic tags and methodology to attack them [17]. The company NXP(who created the MIFARE Chips) tried to cover these publications.

Also, there are well-known cases, like the one made to the London Underground Oyster Card or OV-chipkaart German public transport, allowing the hackers to travel for free.

Attacks such as this suggest that these systems are not as secure as its creators say, and despite the security mechanisms this technology implements, information can be retrieved and altered. If this is true, anyone can get to withdraw money from credit a card wirelessly, use "Bicing" without paying or enter restricted access facilities such as hotels, hospitals and government facilities.

On top there is the appearance in the market of the NFC technology, as seen in the EUSecWest2012 [2] conference; NFC allows to remove the specific hardware and use a smartphone to make the attacks, either to emulate a card to copy or modify the content.

1.4 Viability

1.4.1 Technical viability

All the study and analysis of the security of MIFARE Classic cards is theoretical, and therefore does not depend on other external factors besides the documentation we can find.

The practical application of the project will require specific hardware. The hardware consists of a computer, a device to read/write cards and MIFARE Classic cards themselves from the services to be analysed. An Android smartphone with NFC technology will be required to try to adapt these types of attacks on NFC phones.

For the readers there is a wide range and they can be found easily online.

1.4.2 Economical viability

The cost of the project is based on the human resources and the required hardware for the practical section, that is:

- Computer
- Read/write device for RFID cards
- MIFARE Classic cards
- NFC smartphone

From the previous list of hardware we just lack the tags and the RFID card reader, and we found an OpenSource with a price of about 50€, from there, we can find them for a very wide price range.

1.4.3 Ethical and legal viability

The content of this project will encounter security issues that may affect others, it is very important to make clear that the main aim of this project is not to find vulnerabilities to exploit systems. On the contrary, we want to show security problems to the general public and make them aware of the risks of using this technology. Also, to develop possible corrections to the problems found in the future.

1.5 Memory structure

The memory begins in Chapter 2 with an introduction of the RFID technology and some cryptography concepts. The next chapter is focused in the MIFARE Classic chip structure and also how the cryptographic protocol used in the MIFARE Classic chip works. In Chapter 4 we explain some characteristics that make these chips weak. The next Chapter contains an explanation of the tools used in our tests and Chapter 6 presents our tests and the attacks performed. Finally, we explain some case studies and in the last chapter, the conclusions.

Chapter 2

Basic concepts

In this chapter we present a theoretical approach to the RFID technology and the cryptography used in an attempt of making this kind of systems safe, also introducing important terminology.

2.1 RFID technology

Radio-frequency identification (RFID) is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. This technology is made of two kind of components: the Proximity Integrated Circuit Card (PICC) and the Proximity Coupling Device (PCD). Since the PICCs are not always in card shapes they can be referred to as *transponder* or *tag* which is located on the object or subject to be identified. Also the PCD is commonly referred to as interrogator or *reader*, which, depending on the design, may be a read or write/read device. Those components that are located in the end-user environment constitute the front-end of an RFID system. Optionally the RFID readers can be connected to a back-end system [4].

2.1.1 Classification of RFID systems

We can categorize RFID systems depending on the system structure:

- *online* RFID: In those systems the reader has an online connection to the back-end system, which can be accessed during the transaction. Because of this, it is easier to detect fraud, for example: the multiple use of tickets.

- *offline* RFID: In those systems the reader is offline but contacted in some regular time intervals. Here the readers work autonomously and the transactions are downloaded from time to time by maintenance personnel.

Another way to categorize RFID systems is based on their security functionality:

- *Low-end* RFID Systems: In this category there are low-cost tags. Among these, there are one-bit transponders used for anti-theft in stores and tags with a permanent ID but without any security or privacy functions. Other low-end tags provide some elementary functions: a password-protected KILL command, an access password, read, write and lock memory blocks, checksums (e.g. CRC16) and a pseudo random number generator. The prevalent EPC (Electronic Product Code) Class 1 Generation 2 UHF tags belong to this category.

- *Mid-range* RFID Systems: This category includes read-write transponders of moderate cost with read-write data memory from a few hundred bytes to more than 100 kB. Mid-range RFID systems are used in applications such as ticketing, access control, and automotive immobilizers. Usually, these tags implement mutual authentication and enforce an access control for the transponder's memory. The transmitted data is encrypted and integrity protected over the radio interface. The transponders are usually implemented as a "hard-wired" state machine and often proprietary cryptographic functions are in use. MIFARE Classic is the most prominent example in this category.

- *High-end* RFID Systems: These RFID transponders contain microprocessor-based smartcard chips and are equipped with a smartcard operating system. Both the chip and its operating system implement security mechanisms. The security mechanisms are application specific and include advanced cryptographic functions. In the very high-end, we find dual-interface smartcards with crypto-coprocessors allowing for public key cryptography such as digital signature operation.

There is a set of standards called ISO-14441, ISO-14442, ISO-14443 and ISO-14444 that define communication protocols and data exchange formats in RFID. Near Field Communication (NFC) is a RFID data interface between devices, similar to Bluetooth. NFC is a subset of the RFID technology and its characteristics are of interest in relation to smartphones and smartcards as it guarantees their interaction as we will see in section 2.1.4.

2.1.2 Operative frequencies and range

Different requirements in the RFID systems make necessary to use a wide range of frequencies. On the one hand, the systems using frequencies between 100kHz and 30MHz operate using inductive coupling, on the other hand, the systems using frequencies between 2.45GHz and 5.8GHz are coupled using electromagnetic fields[9]. Microwave systems have a higher range, more memory capacity and are less sensitive to electromagnetic interference fields. However, in contrast to inductive systems, microwave systems require a backup battery. Inductive systems can operate using the energy provided by the reader transmission.

- Low Frequency (LF): Typically operates at 125kHz or 134kHz. This frequency band provides a short read range (<0.5m) and a slow read speed, but tend to be less sensitive to interference.
- High Frequency (HF): These systems operate at 13.56MHz, and perform a greater read range (<1.0m) and higher read speed than LF tags. Also, the price is among the lowest of all RFID tags, and tend to be more sensitive to interference than LF.
- Ultra High Frequency (UHF): Utilizes the 860 to 930MHz band (typically 868MHz in Europe). This frequency band range is up to 10.0m and the data transfer rate is faster than HF. UHF lacks the ability of read tags through water or metal.
- Microwave: This band operates either in 2.45Ghz or 5.8GHz. It provides the highest data reading rate and has the highest reading range (>3.0m). They are the most expensive tags and are not able to penetrate objects with high water or metal content.

	LF	HF	UHF	Microwave
Frequency	<135kHz	13.56MHz	860/915MHz	2.45GHz
Range	<0.5m	<1m	<10m	>3m

Table 2.1: RFID ranges and frequencies

2.1.3 Applications

RFID technology can be used in multiple applications:

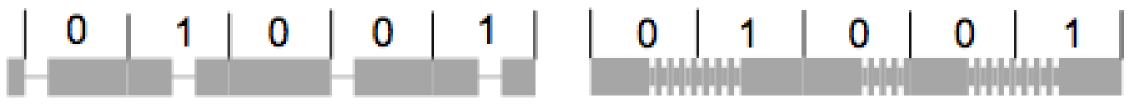
- EPC: The Electronic Product Code is a universal product identifier made as a substitute of the bar code. They are low-end tags and use the UHF range.
- Contactless Payment Systems: They are used in credit cards and smartphones, they usually operate at HF and they should always be high-end tags.
- Electronic Passport: It includes a Mid-range RFID chip operating at HF including the owners information and photo.
- Access Control: Depending on the security requirements these tags can be from low to high-end and usually operate at HF.
- Animal Identification: Equivalent to EPC but operating at LF, it is used for live stock, it can also be used to store information from domestic animals.
- Electronic Immobilisation: It is integrated in car keys to make them harder to copy. The car only starts if you use the right key and it includes the correct RFID chip. It uses low-end tags and operates at LF.
- Sporting Events: It helps to avoid fake tickets and illegal resale by including a RFID chip with the serial number. It usually uses low-end tags transmitting at HF for the low cost.
- Medical Applications: With different kinds of sensors, temperature, blood pressure, glucose and other biological parameters can be monitorized for medical purposes. The most common use is tracking of medical supplies with low-end tags operating at HF.

2.1.4 NFC

The NFC is a communication protocol made as a subset of RFID technology. It operates at HF (13.56MHz) and includes all devices from Low to High-end, among these devices we can find tags and also smartphones. The ISO-14441, ISO-14442, ISO-14443 and ISO-14444 are the principal standards that define all the protocol layers from physical to application to guarantee the interoperability of all kinds of devices regardless of the manufacturer.

According to the ISO-14443-A standard each component uses a different encoding to communicate. The reader uses the Modified Miller, and the tag uses the Manchester encoding.

The Modified Miller encoding has a low pulse at the beginning of the bit to encode a 0, to encode a 1 the low pulse is delayed half a period. And if a 0 is preceded by a 1 the 0 is encoded with no low pulse as seen in Figure 2.1(a). Figure 2.1(b) shows the Manchester encoding, in this case the bit duration is split in two halves. A 0 is produced by introducing field resistance in the second half of the bit duration while a 1 is encoded by introducing field resistance in the first half.



(a) Modified Miller (b) Manchester

Figure 2.1: Encoding methods used by reader and tag

Other significant characteristics are the reading range limited to 10cm and the timing limitations made to enhance the security and reliability. The MIFARE Classic tags (which we will see in Chapter 3) from NXP claim to be NFC compatible but in some of the newest smartphones it is not supported as these tags use a proprietary "secret" protocol.

2.2 Cryptography

Cryptography is the science and practice of hiding information [23] and there are several methods to do it. In this section we will approach the techniques that are used in the MIFARE Classic chips.

2.2.1 Symmetric encryption

The symmetric encryption basically consists of 4 elements: message, key, encryption algorithm and decryption algorithm. This method is used to transform a plaintext into a ciphertext and vice versa with the same key. To encrypt, the algorithm receives the plaintext and the key as input, and returns the ciphertext. Using the same key, the ciphertext and the decryption algorithm, the plaintext is recovered.

In general, this method is fast even with large amounts of data. The two most remarkable limitations are the key exchange and the number of keys to store, the reason is that for each group sharing a secret a different key is needed.

In Chapter 3 we will see how the MIFARE Classic uses symmetric encryption in the communication between the reader and the tag.

2.2.2 Stream ciphers

Stream ciphers are a very fast kind of symmetric encryption algorithm which works ciphering individual characters or bits using a transformation. In our case this transformation is accomplished by XORing the message with a pseudo-random sequence, this is done both to encrypt and decrypt as we can see in Figure 2.2. The pseudo-random sequence is created trough a pseudo-random number generator using the key as seed [23].

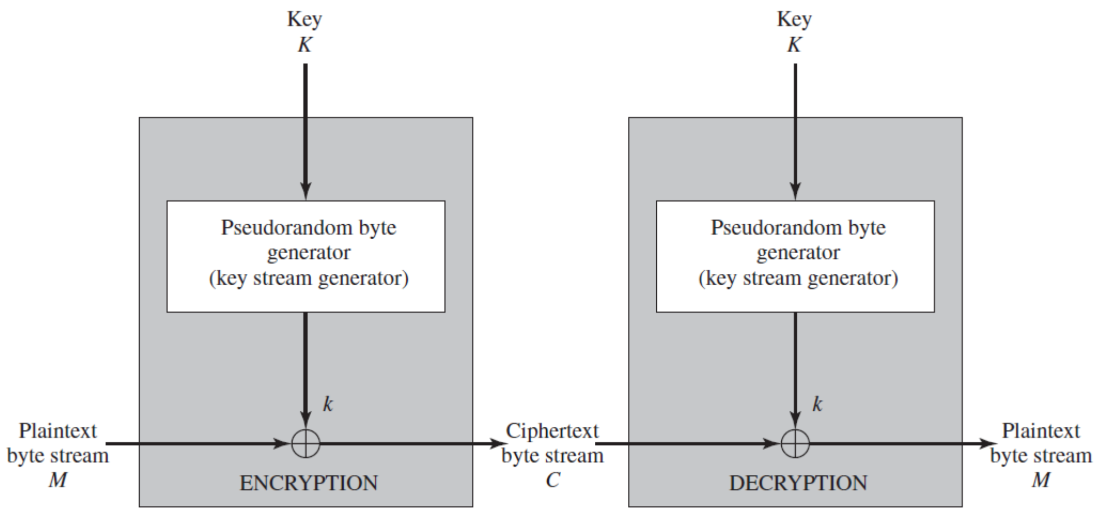

Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

Figure 2.2: Structure of stream ciphers

When a sequence of random numbers is needed the best we can do is use very long period sequences, called pseudo-random sequences. For a periodic sequence to be considered pseudo-random it must fulfil the Golomb principles:

1. The number of zeros and ones should be as equal as possible per period.
2. Half of the runs (bursts) in a cycle have length 1, one quarter have length 2, one eighth have length 3, and so forth. Moreover, half of the runs of a certain length are gaps(...10001...), the other half are blocks(...01110...).
3. The out-of-phase autocorrelation AC(p) is two-valued, take the value 1 if p is a multiple of the period N and another constant value otherwise.

Moreover the sequence must have a very long period, be easy to generate and cryptographically secure against chosen plaintext attack.

A common implementation of pseudo-random number generators is the Linear Feedback Shift Registers (LFSR) which is a shift register whose input bit is a linear function (XORs in our case) of its previous state [4]. After the initialization of the registers each extracted bit makes all the registers shift and triggers the XORing that determines the value of the input register. The principal weakness of the LFSRs is that with enough data from the output it can be reverse-engineered; a usual solution is to apply non-linear functions to the LFSR output, this way it is more difficult to recover the LFSR state. In Figure 3.7 we can see an LFSR with some filter functions.

In the context of the project, the MIFARE Classic chip uses a stream cipher and generates the pseudo-random numbers with a LFSR and filter functions [24]. We will focus on this in Chapter 3.

2.2.3 Challenge-Response Authentication

Challenge-response authentication is an authentication mechanism in which a system presents a question, known as *challenge*, and the user must provide a valid answer, known as *response*, to be authenticated. The implementations for this system involve mutual authentication, where the different parts must convince each other that they know the password without it being transmitted.

To achieve mutual authentication each part of the communication can use the password as encryption key to encrypt and transmit random-generated data as challenge so the response to this challenge must be the decrypted data. These data is usually known as *nonce*. Using this method we can avoid the password eavesdropping. Also, the usage of random-generated data decreases the possibility of a replay attack, where the attacker eavesdrops the communication to retransmit it later attempting to authenticate. Nonces should be randomly generated to avoid attacks.

As we will see in Chapter 3, the MIFARE Classic chip performs a mutual challenge-response authentication using the LFSR to pseudo-randomly generate nonces.

Chapter 3

MIFARE Classic tags

In this chapter we examine the company's RFID NXP Semiconductors chip, MIFARE Classic. Given that this tag model holds the 80% of the market applications, we believe it is a significant case of study because of its implications from the point of view of security. The MIFARE Classic chip has two versions 1K and 4K, apart from the memory size the diference lies in the memory organization. We will describe the 1K Classic chip in detail.

3.1 Structure Overview

The MIFARE Classic chip is an EEPROM memory with some cryptographic features. According to the data sheets from NXP the basic unit of the memory is a 16-byte data block. In the 1K card, data blocks are grouped together to form 16 sectors of 4 data blocks each. Data blocks in the 4K card are distributed in 32 sectors of 4 blocks each and 8 sectors of 16 blocks each [22]. The logical structure of the chip is shown in Figure 3.1.

In the MIFARE Classic 1K the block 0 from sector 0 is write protected, it holds the Unique Identification Number (UID) and manufacturer data, as we can see in Figure 3.2, and it also has a bit count check (BCC) which is the result of XORing the UID bytes. Each sector has 4 blocks, the first 3 are for storing data (except in the block 0 case) and the last one is known as the sector trailer.

The first three data blocks of each sector (except block 0) can be configured into read/write mode or value mode. The read/write mode allows to store raw data in the block and the value mode is for electronic purse functions. Value blocks have a fixed data format which permits error detection, correction and a backup management. This format consists in writing the same data three times, one inverted and two non-inverted.

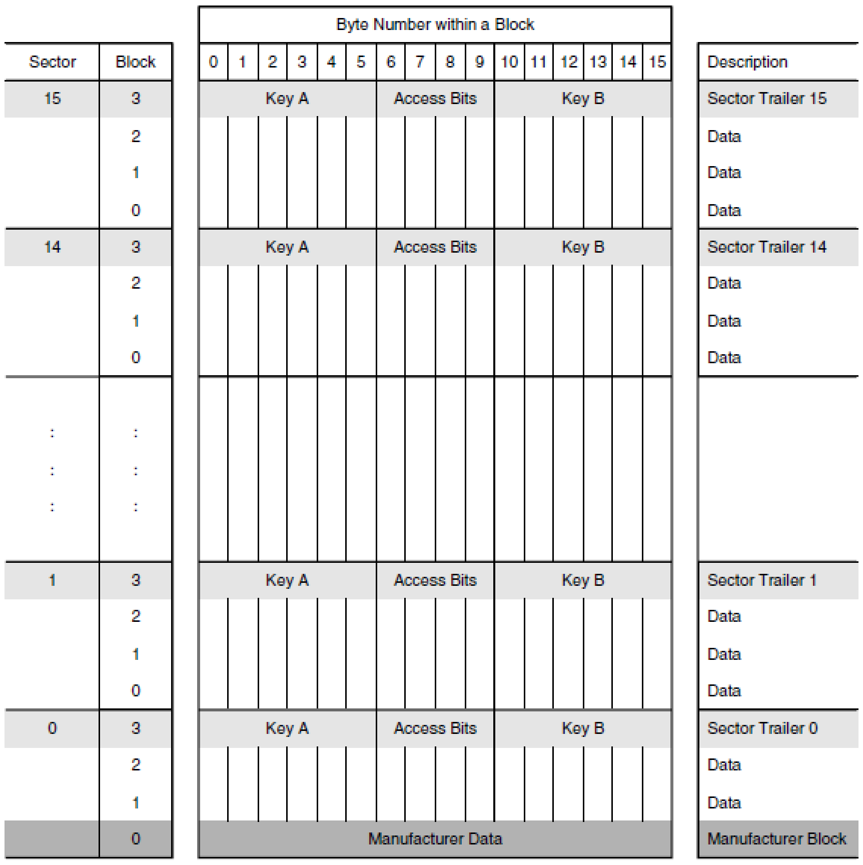


Figure 3.1: MIFARE Classic 1K memory structure

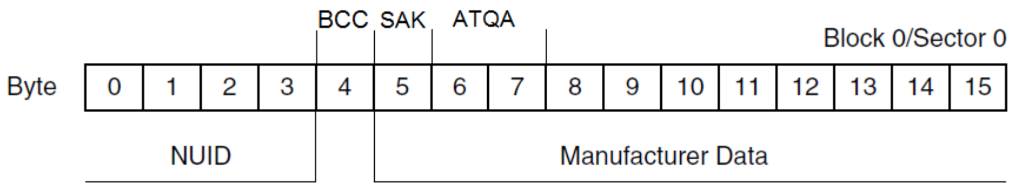


Figure 3.2: MIFARE Classic 1K manufacturer block

The sector trailer is used to store the keys of 48 bits each and the permissions of the sector containing it. These sector trailers contain the A key (bytes 0 to 5), the access bits (bytes 6 to 9) and the B key (bytes 10 to 15). The reader needs to be authenticated by a sector before any memory operations are allowed.

After authentication any of the operations shown in Table 3.1 can be done:

Read	reads one block
Write	writes one block
Decrement	decrements the contents of one block and stores the result in the data-register
Increment	increments the contents of one block and stores the result in the data-register
Transfer	writes the contents of the data-register to one block
Restore	stores the contents of one block in the data-register

Table 3.1: Memory Operations

3.2 Permissions

Each sector trailer contains keys A an B and 3 bits for each data block in that sector. These bits define the access conditions of the blocks of that sector. The bits are stored twice, once inverted and once non-inverted making a total of 3 bytes plus 1 blank byte as we can see in Figure 3.3, with the purpose of performing error detection and error correction. These bits control the memory access rights to the sector trailers and the data blocks, including the interaction with keys A and B. The access conditions for sector trailers and data blocks are shown in Tables 3.3 and 3.2 respectively.

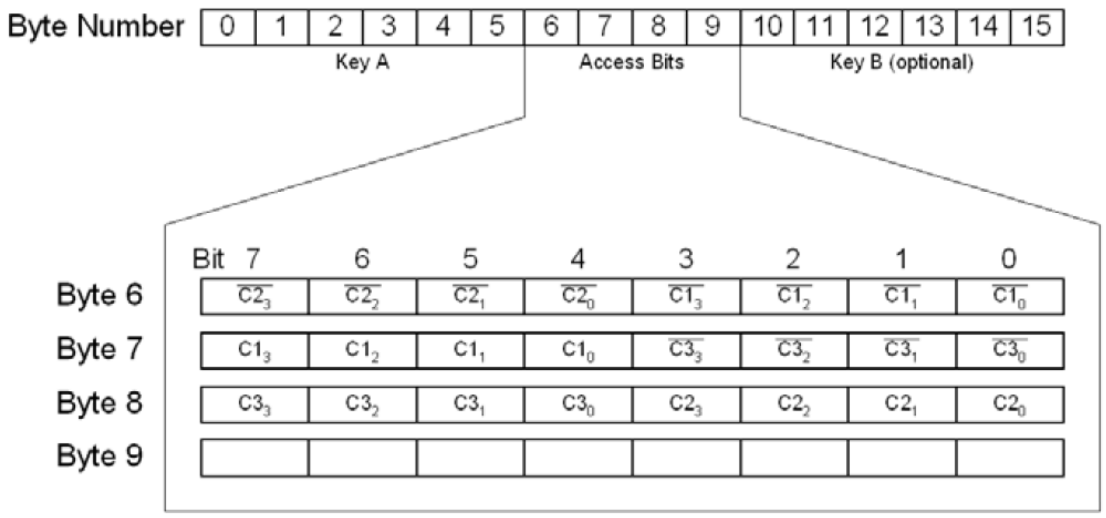

Byte Number
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Key A
Access Bits
Key B (optional)
Bit 7 6 5 4 3 2 1 0
Byte 6
$\overline{C2_3}$ $\overline{C2_2}$ $\overline{C2_1}$ $\overline{C2_0}$ $\overline{C1_3}$ $\overline{C1_2}$ $\overline{C1_1}$ $\overline{C1_0}$
Byte 7
$C1_3$ $C1_2$ $C1_1$ $C1_0$ $\overline{C3_3}$ $\overline{C3_2}$ $\overline{C3_1}$ $\overline{C3_0}$
Byte 8
$C3_3$ $C3_2$ $C3_1$ $C3_0$ $C2_3$ $C2_2$ $C2_1$ $C2_0$
Byte 9

Figure 3.3: MIFARE Classic access conditions

As an example, for the fixed values of $C1_i = 1$, $C2_i = 0$ and $C3_i = 0$ (where i is a value from 0 to 3 that indicates the block) on an ordinary block (i between 0 and 2) only read and write operations will be allowed, the read operation will be possible by authenticating with keys A or B but the write operation will be only permitted providing the B key. On the other hand if these permission values refer to the sector trailer ($i = 3$) it will mean that it is not permitted to overwrite the access conditions but we will be able to read them

C1	C2	C3	read	write	incr	decr, transfer, restore
0	0	0	key A|B	key A|B	key A|B	key A|B
0	1	0	key A|B	never	never	never
1	0	0	key A|B	key B	never	never
1	1	0	key A|B	key B	key B	key A|B
0	0	1	key A|B	never	never	key A|B
0	1	1	key B	key B	never	never
1	0	1	key B	never	never	never
1	1	1	never	never	never	never

Table 3.2: Access conditions for data blocks $(i = 0, 1, 2)$

C1	C2	C3	read	write	read	write	read	write
0	0	0	never	key A	key A	never	key A	key A
0	1	0	never	never	key A	never	key A	never
1	0	0	never	key B	key A|B	never	never	key B
1	1	0	never	never	key A|B	never	never	never
0	0	1	never	key A	key A	key A	key A	key A
0	1	1	never	key B	key A|B	key B	never	key B
1	0	1	never	never	key A|B	key B	never	never
1	1	1	never	never	key A|B	never	never	never

Table 3.3: Access conditions for sector trailer $(i = 3)$

by authenticating with any of the A or B keys and these keys will be overwrittable only with previous authentication using the B key.

3.3 Communication

Figure 3.4 shows the communication flow of the MIFARE Classic. In the following sections we will explain the phases of this process and its abbreviations.

3.3.1 Request

The tag starts at the Power On Reset (POR) state and the reader sends repeated request standard (REQA) or request all (WUPA) commands, then the tags can respond by sending the answer to request code (ATQA). As we can see in Table 3.4 the ATQA code is related to the tag model and manufacturer. At this point the reader recognises that at least one card is in the read range.

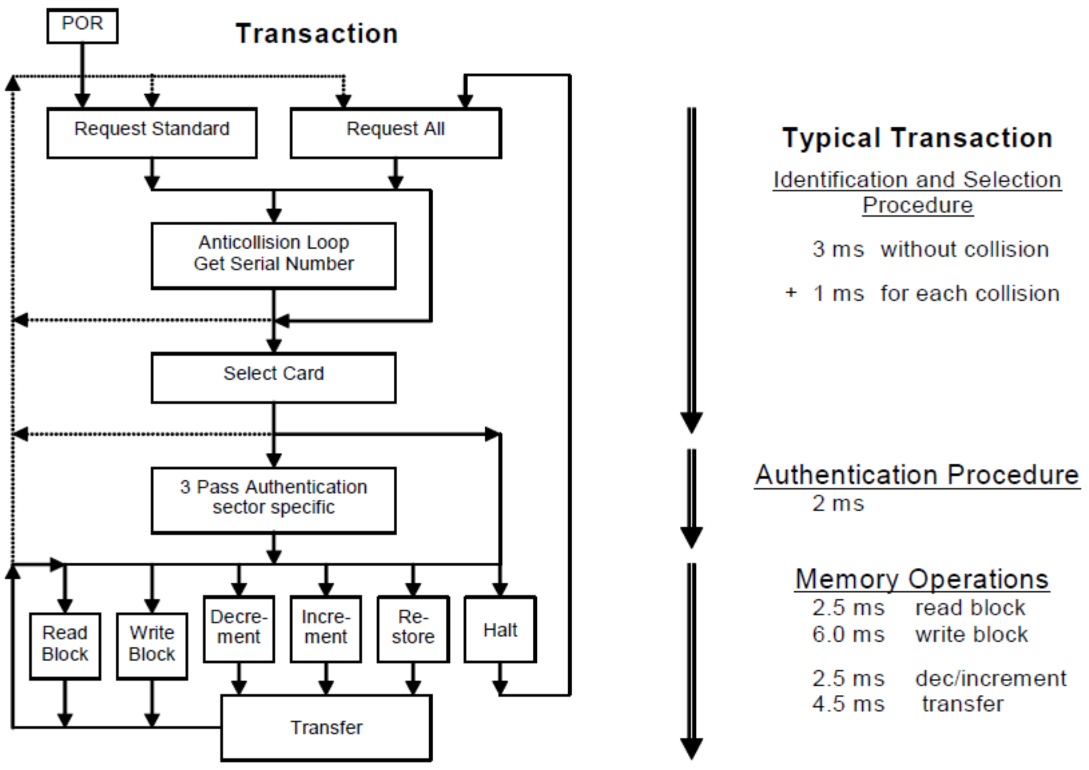


Figure 3.4: MIFARE Classic transaction flow.

3.3.2 Anticollision Loop

In the anticollision loop an Anticollision command (AC) is sent, then the serial number of a card with its BCC is received. If there are multiple tags in the reader range, a collision occurs. The reader uses only the valid received values to send another Anticollision command updated with the new data, then only the tags whose UID match will respond. This is repeated until there is no collision. This way the tags can be distinguished by their UID.

Manufacturer	Product	ATQA	SAK
NXP	MIFARE Mini	04 00	09
	MIFARE Classic 1k	04 00	08
	MIFARE Classic 4k	02 00	18
	MIFARE Ultralight	44 00	00
	MIFARE DESFire	44 03	20
	MIFARE DESFire EV1	44 03	20
Infineon	MIFARE Classic 1k	04 00	88

Table 3.4: MIFARE Classic ATQA and SAK values

3.3.3 Card Selection

The reader selects one individual card with the select card command (SEL) and the card returns the select acknowledge reply (SAK) code which is meant to determine the type of the selected card.

3.3.4 Authentication

After the tag selection, the reader specifies the memory location of the next memory access and uses the corresponding key for the 3-step authentication procedure. After a successful authentication all memory operations are encrypted as we will see in Section 3.4.

At any point of the communication the reader can stop the communication by sending a halt command (HLTA), then the tag will go into a halt state and only will be able to awake with a WUPA command. More detailed information and the command set of MIFARE Classic can be found in Garcia's article [7].

3.3.5 Example trace

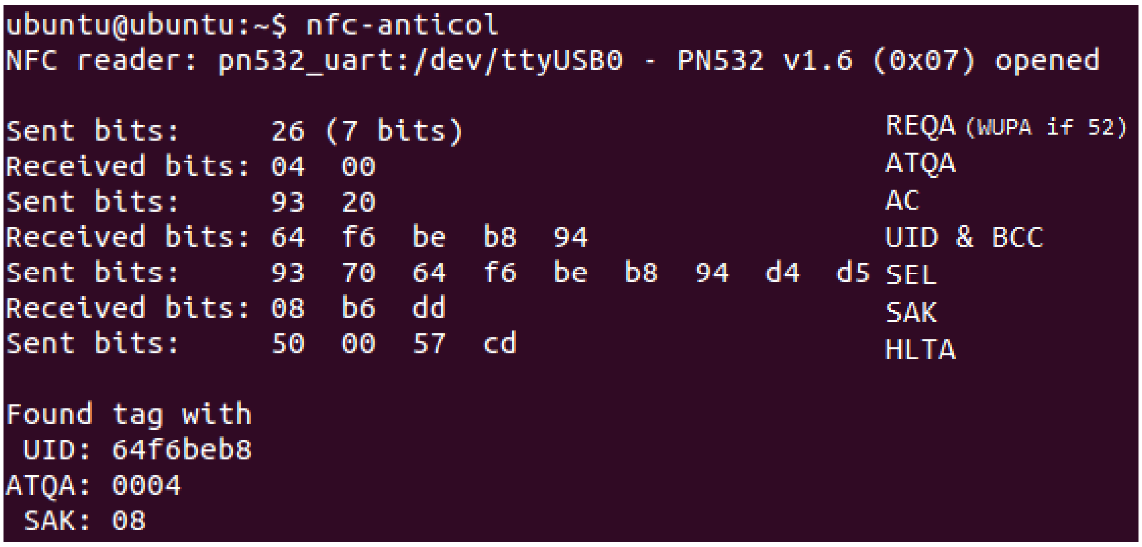
In Figure 3.5 we can see a trace of a reader initial communication from the request to the card selection where the values of the different commands are shown.

The communication starts with the reader sending a request command (REQA) and the tag responds with the ATQA code, after that the anticollision loop starts with the anticollision command (AC) then the tag responds with the UID and an error check code (BCC). At this point the card selection phase starts with the SEL and SAK commands. Finally, the reader stops the communication with a HLTA command.

In Table 3.5 another communication trace is shown including the authentication process and data transmission of block 0.

3.4 The cryptographic algorithm: Crypto1

Crypto1 is the algorithm used in MIFARE Classic chips for encryption. Developed by NXP Semiconductors, it's proprietary and bases part of its security on the secrecy of the Crypto1 algorithm. Despite the secrecy, Nohl and Plötz uncovered the algorithm by reverse engineering the chip implementations [18], also Garcia *et.al.* disclosed the entire



```
ubuntu@ubuntu:~$ nfc-anticol
NFC reader: pn532_uart:/dev/ttyUSB0 - PN532 v1.6 (0x07) opened

Sent bits:     26 (7 bits)                        REQA (WUPA if 52)
Received bits: 04  00                             ATQA
Sent bits:     93  20                             AC
Received bits: 64  f6  be  b8  94                 UID & BCC
Sent bits:     93  70  64  f6  be  b8  94  d4  d5 SEL
Received bits: 08  b6  dd                         SAK
Sent bits:     50  00  57  cd                     HLTA

Found tag with
 UID: 64f6beb8
ATQA: 0004
 SAK: 08
```

Figure 3.5: Example of communication

algorithm [12].

In Figure 3.6 we can see the basic structure of the Crypto1 and we will describe in detail the relevant components in the following subsections.

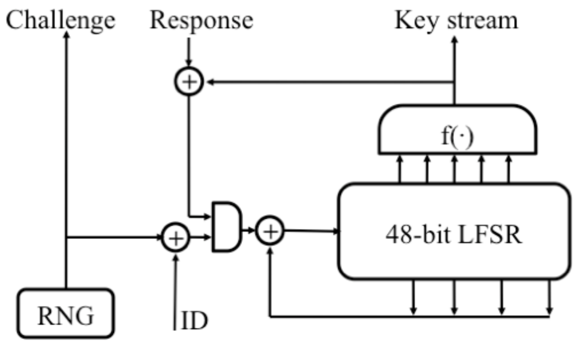


Figure 3.6: Structure overview of Crypto1

3.4.1 Data transmission

All the memory operations on the MIFARE Classic tag are transmitted encrypted using Crypto1 as a stream cipher. The pseudo-random number generator used for this purpose, as we can see in Figure 3.7, is a 48-bit LFSR with a two-layer non-linear filter.

At each clock cycle, a keystream bit is generated by applying the filter functions to twenty bits from the LFSR. Then the LFSR shifts to the left discarding the 0th bit and inserts a

	Sender	Bytes	Command
1	PCD	26	REQA (WUPA if 52)
2	PICC	04 00	ATQA
3	PCD	93 20	AC
4	PICC	9C 59 9B 32 6C	UID & BCC
5	PCD	93 70 9C 59 9B 32 6C 6B 30	SEL
6	PICC	08 B6 DD	SAK
7	PCD	60 00 F5 7B	Authenticate block 0 using key A
8	PICC	82 A4 16 6C	Tag nonce
9	PCD	EF EA 1C DA 8D 65 73 4B	Reader nonce & response
10	PICC	9A 42 7B 20	Tag response
11	PCD	30 00 02 A8	READ
12	PICC	9C 59 9B 32 6C 88 04 00 47	DATA
		C1 2D 2A C9 00 28 07 5A 41	
13	PCD	50 00 57 cd	HLTA

Table 3.5: Complete communication trace

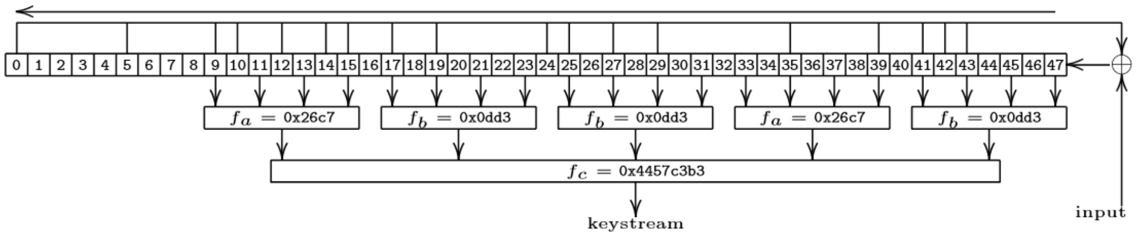


Figure 3.7: Crypto1's LFSR and filter functions

new bit on the right using the feedback function. This feedback function can be defined by:

$$L(x_0x_1...x_{47}) = x_0 \oplus x_5 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{19} \oplus x_{24} \oplus x_{25} \oplus x_{27} \oplus x_{29} \oplus x_{35} \oplus x_{39} \oplus x_{41} \oplus x_{42} \oplus x_{43} \tag{3.1}$$

The filter that generates one bit out of the 20 bits extracted from the LFSR is defined by a combination of the functions f_a, f_b and f_c as we can see in (3.5). Each one of the filter functions are defined in Equations (3.2), (3.3) and (3.4).

$$f_a(a, b, c, d) = ((a \vee b) \oplus (a \wedge d)) \oplus (c \wedge (a \oplus b) \vee d) \tag{3.2}$$

$$f_b(a,b,c,d) = ((a \wedge b) \vee c) \oplus ((a \oplus b) \wedge (c \vee d)) \tag{3.3}$$

$$f_c(a,b,c,d,e) = (a \vee ((b \vee e) \wedge (d \oplus e))) \oplus ((a \oplus (b \wedge d)) \wedge ((c \oplus d) \vee (b \wedge e))) \tag{3.4}$$

$$\begin{aligned} f(x_0x_1...x_{47}) = f_c(&f_a(x_9,x_{11},x_{13},x_{15}), f_b(x_{17},x_{19},x_{21},x_{23}), f_b(x_{25},x_{27},x_{29},x_{31}),\\ &f_a(x_{33},x_{35},x_{37},x_{39}), f_b(x_{41},x_{43},x_{45},x_{47})) \end{aligned} \tag{3.5}$$

The ISO-14443-A standard specifies that every byte sent is followed by a parity bit for error detection, so the MIFARE Classic sends a parity bit with each byte transmitted but calculates it over the plaintext instead of the ciphertext and then encrypts it. Moreover the parity bit is encrypted with the same value as the next bit transmitted.

3.4.2 Initialisation and Authentication

The authentication protocol used in MIFARE Classic follows a mutual challenge-response mechanism. In this case, the challenge is a 32-bit nonce generated by a 16-bit LFSR, a different LFSR from the 48-bit LFSR used for the cipher.

During each clock cycle, the 16-bit LFSR generates one bit and inserts a new rightmost bit using the feedback function defined by Equation (3.6).

$$L(x_0x_1..x_{15}) = x_0 \oplus x_2 \oplus x_3 \oplus x_5 \tag{3.6}$$

According to the design of the protocol, for a given 32-bit sequence, the next sequence can be computed using the successor function (3.7), then suc^n is the result of applying n times the suc function as defined by the Equation (3.8).

$$suc(x_0x_1...x_{31}) = x_1x_2...x_{31}L(x_{16}x_{17}...x_{31}) \tag{3.7}$$

$$suc^n(x_0x_1...x_{31}) = suc(suc^{n-1}(x_0x_1...x_{31})) \tag{3.8}$$

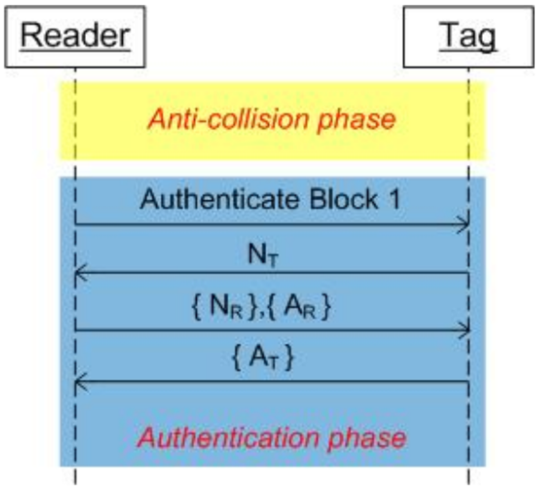


Figure 3.8: Authentication phase

The authentication process shown in Figure 3.8 can be described as:

1. The reader sends an authentication request to the tag.

2. The tag sends a challenge nonce N_T. The next messages will be encrypted (XORing with the keystreams $ks1$, $ks2$ and $ks3$ respectively which we will explain afterwards).

3. The reader replies with an encrypted[1] nonce N_R and an encrypted response to the previous challenge A_R.

4. If the reader response is correct, the tag will respond with an encrypted answer A_T, otherwise the tag will not respond.

5. Once the reader has checked correctness of the tag response the authentication is concluded and operation commands can be done.

The calculation of the challenge responses was discovered by Garcia [11] who also discovered that the tag nonce depends only on the time between the power-up an the start of the communication.

Given 32-bit sequence the next sequence can be computed using the successor function $suc(N_T)$, so the answer to the nonces are calculated as shown in Equations 3.9 3.10 an encrypted with the corresponding keystreams before they are sent.

[1]Keys { } denote encrypted values in Figure 3.8. The detailed encryption procedure is described next

$$A_R = suc^{64}(N_T) \tag{3.9}$$

$$A_T = suc^{96}(N_T) \tag{3.10}$$

In order to synchronize the 48-bit LFSRs, the stream ciphers have to be initialised at the reader and tag during the authentication session. The parameters necessary for this process are the 48-bit key K the nonces N_T and N_R, and the tag UID, where

$$K = k_0 k_1 k_2 ... k_{47} \tag{3.11}$$

$$N_T = n_{T,0} n_{T,1} n_{T,2} ... n_{T,31} \tag{3.12}$$

$$UID = u_0 u_1 u_2 ... u_{31} \tag{3.13}$$

$$N_R = n_{R,0} n_{R,1} n_{R,2} ... n_{R,31} \tag{3.14}$$

and the internal cipher state at time i (S_i), which is the values of the LFSR at time i, are defined as

$$S_i = s_i s_{i+1} s_{i+2} ... s_{i+47} \tag{3.15}$$

Now we are going to explain the authentication process in greater detail including the initialisation of the LFSR. As we can see in Figure 3.9 authentication process starts with the initialization of the 48-bit LFSR and ends achieving a state of s_i which will allow a communication between the tag and the reader. It is defined by the following steps:

1. First the reader sends an authentication request indicating the block to read.

2. Then the reader and tag ciphers are initialized with the 48-bit sector key

$$s_i = k_i, \forall_i \in [0, 47] \tag{3.16}$$

3. Next, the tag sends the nonce N_T from the 16-bit LFSR to the reader.

4. Then $N_T \oplus UID$ is shifted in the reader LFSR with the feedback

$$s_{i+48} = L(s_i, s_{i+1}, ..., s_{i+47}) \oplus n_{T,i} \oplus u_i, \forall_i \in [0, 31] \tag{3.17}$$

5. The reader picks a nonce N_R and uses the keystream ($ks1$) obtained in the previous operation to cipher it. Then, the reader computes the A_R using the successor function (as seen in Equation (3.9)), after that, the N_R is shifted in the reader LFSR and the keystream generated ($ks2$) is used to encrypt the A_R. Finally, the reader sends N_R, A_R to the tag.

$$s_{i+80} = L(s_{i+32}, s_{i+33}, ..., s_{i+79}) \oplus n_{R,i}, \forall_i \in [0, 31] \tag{3.18}$$

6. The tag decrypts the N_R using $ks1$, then N_R is shifted in the tag LFSR and the keystream generated ($ks2$) is used to decrypt A_R. Finally the tag calculates the A_T using the successor function (as seen in Equation (3.10)), and a new keystream is generated ($ks3$) to encrypt before sending.

7. The reader generates a keystream ($ks3$) to decrypt the A_T and check the correctness of the authentication.

8. At this point the tag and the reader have the same state and from now on the stream cipher can be used

$$s_{i+112} = L(s_{i+64}, s_{i+65}, ..., s_{i+111}), \forall_i \in N \tag{3.19}$$

Next, we formalize some of the concepts seen in the initialisation and authentication process. We can define the keystream bit b_i at time i, obtained from the LFSR using the function of Equation (3.5), as

$$b_i = f(s_i s_{i+1} ... s_{i+47}), \forall_i \in N \tag{3.20}$$

The encryption of N_R, A_R and A_T used in the communication of the 3-step authentication is denoted using { } as we can see in the next equations.

$$\{n_{R,i}\} = n_{R,i} \oplus b_{i+32}, \forall_i \in [0, 31] \tag{3.21}$$

$$\{a_{R,i}\} = a_{R,i} \oplus b_{i+64}, \forall_i \in [0, 31] \tag{3.22}$$

$$\{a_{T,i}\} = a_{T,i} \oplus b_{i+96}, \forall_i \in [0, 31] \tag{3.23}$$

For references we also group the previous keystream bits together using the following common notation.

$$ks1 = b_{32}b_{33}...b_{63} \tag{3.24}$$

$$ks2 = b_{64}b_{65}...b_{95} \tag{3.25}$$

$$ks3 = b_{96}b_{97}...b_{127} \tag{3.26}$$

In Figure 3.9 a graphical description of the process can be seen. There, we can see the initialisation phases in both components (reader and tag).

For the understanding of the figure we define the function $word()$ which returns a 32 bit word from the 48-LFSR cipher and the arguments define what is being shifted in while extracting this word.

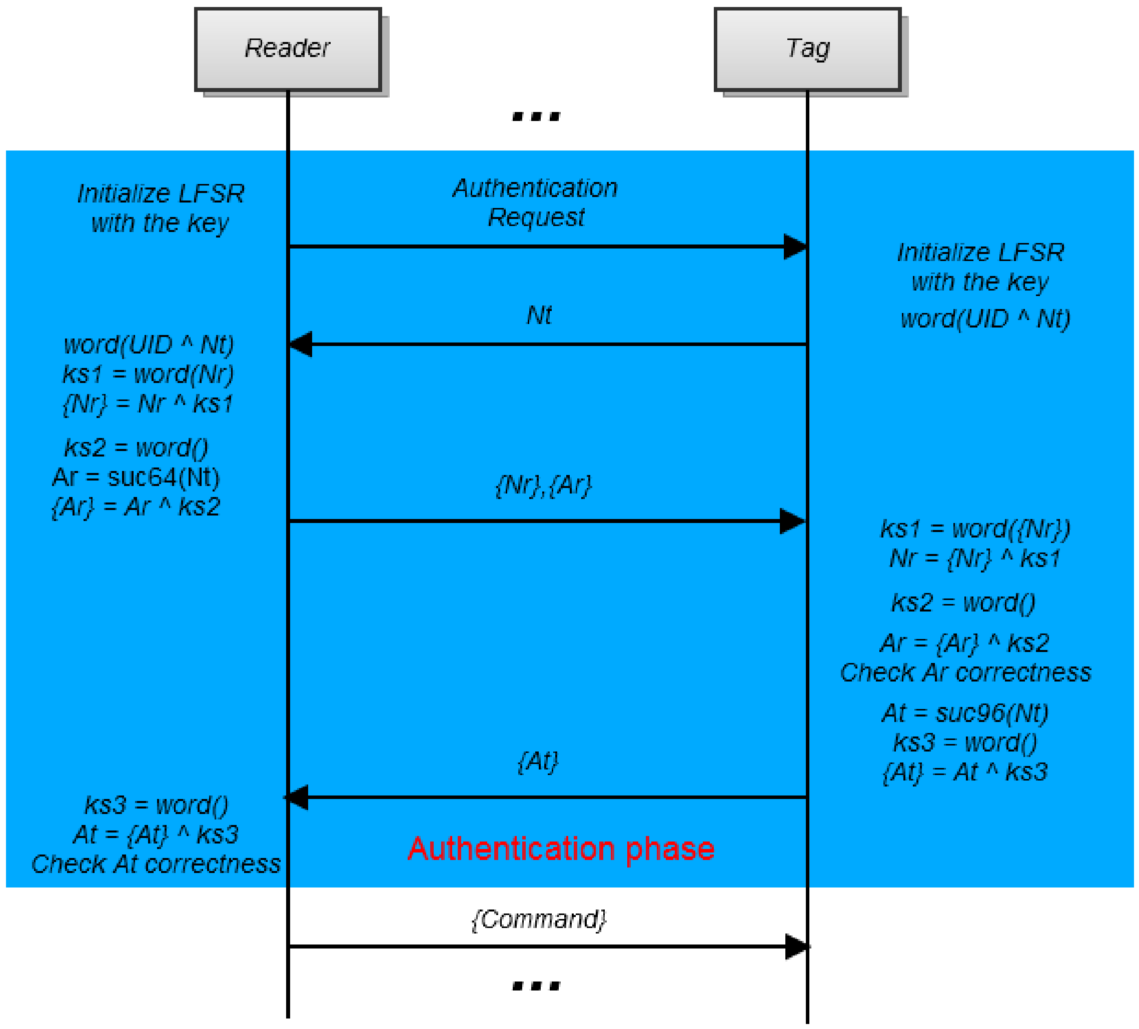


Figure 3.9: Detailed Authentication Phase

Chapter 4

Vulnerabilities of the MIFARE Classic tags

In this chapter, we will examine and explain some weaknesses of the Crypto1 protocol and the MIFARE Classic tag implementation based on the documentation from NXP, the work done by other researchers and also from practical experimentation.

These vulnerabilities can be found either in the pseudo-random number generator, the cryptographic cipher, the communication protocol and the implementation of the system.

4.1 Pseudo-random number generator weaknesses

4.1.1 Low entropy of random generator

From the Equation (3.6) we can see that the pseudo-random number generator used to create the nonces is only 16-bit wide, then the LFSR generated sequence has a period of $2^{16} - 1$ (65535). From previous investigations [17] we know that it takes only 0.6 seconds for the 16-bit generator to generate all the possible nonces. Also the LFSR resets to a known state every time it powers up.

4.2 Stream cipher weaknesses

4.2.1 Odd state bits in LFSR feedback

According to the Equation (3.1), the LFSR state bits used in the filter function are some of the odd-numbered bits. The odd 9^{th} to 47^{th} bits are used to generate a keystream bit.

Then, after two shifts of the LFSR the original 20 bits used for the input of the filter function are shifted to the left, so that the 9^{th} bit is discarded and a new 47^{th} bit is added. The next keystream bit is generated by using these bits. If we know two consecutive keystream bits, we can narrow down the possibilities for the state bits of the 48-bit LFSR [24], a similar process can be used for the even bits.

4.2.2 Leftmost LFSR bits not used in the stream cipher filter

As it can be seen in Figure 3.7, we can see that the bits from positions 0 to 8 are not used in the filter functions. This allows us to perform a reverse filter function to “roll-back” the LFSR state to recover the secret key. The “roll-back” process consists of doing the inverse operation applied to the LFSR during the Authentication.

If we know the LFSR state immediately after N_R has been fed into the streamcipher (as seen in Section 3.4.2) the process consists in “unshifting” the LFSR to the right, then the rightmost bit(the 47th) will be discarded and a new 0^{th} bit r will be inserted (note that the value of the new bit is irrelevant at this point as we don’t need it in the filter functions), now that we have undone one step of the LFSR we can proceed to the next step.

Next we use the filter functions to compute the keystream bit used to encrypt the bit $n_{R,31}$. With the keystream bit obtained we perform an XOR with $\{n_{R,31}\}$ to recover $n_{R,31}$. At this point we can use it to set r to the correct value using the feedback function of the LFSR (Figure 4.1).

Repeating this procedure 31 more times we obtain the LFSR state before N_R is fed in, and 32 more times to recover the secret key before $UID \oplus N_T$ is shifted in.

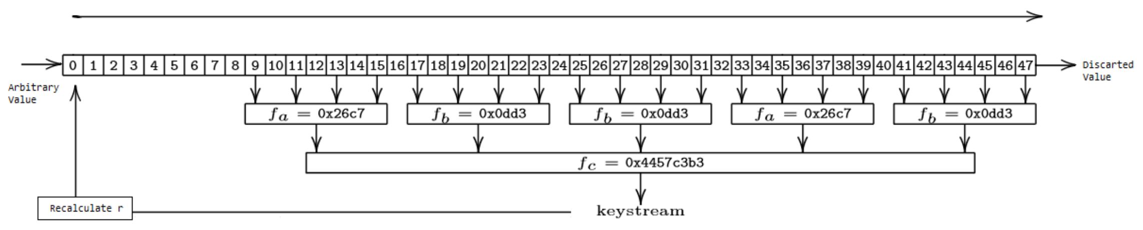


Figure 4.1: Unshifting the Crypto1’s LFSR

4.2.3 Statistical bias in cipher

The 48-bit LFSR output differs from a true random sequence. Comparing the output with a sequence from RANDOM.ORG [13] we can apreciate that it may have some kind of bias.

Courtois investigated on the impact of varying a few bits of N_R for the generation of keystream $ks1$ with Crypto1. In his experiments, he fixed the first three bytes of N_R and varied only the last bits of N_R. His results showed that, with a probability of 0.75, the keystream $ks1$ is independent of the last three bits of N_R, whereas, in fact, the probability should ideally be 0.5. He also made use of this weakness in his card-only attack in [6].

4.3 Protocol weaknesses

4.3.1 Reuse of parity bits

The ISO-14443-A standard defines that every byte transmitted is followed by a parity bit. However, the MIFARE Classic calculates the parity bit over the plaintext instead of the ciphertext. So the 48-bit LFSR does not shift when the parity bit is encrypted. This means that both the parity bit and the first bit of the next plaintext byte are encrypted with the same keystream bit (see Figure 4.2). This violates the property whereby a one-time pad should not be reused.

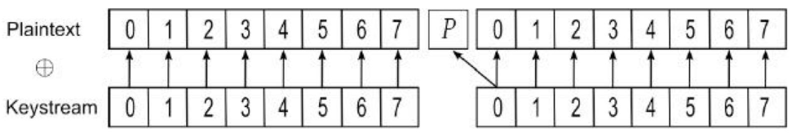


Figure 4.2: One-time pad reuse

4.3.2 Information leak from error code

During the authentication process when the reader sends N_R and A_R the tag checks the parity bits before checking if A_R is correct. If any of the eight parity bits is incorrect the tag does not respond, but if all parity bits are correct and A_R is wrong the tag will respond with a 4-bit error code `0x5`. This error code is also sent encrypted, so XORing the message with the error code `0x5` we can recover four keystream bits.

4.3.3 Multiple sector authentication

After a successful authentication the LFSR is used to generate the keystreams to encrypt the communications, this encrypted session is kept for all kind of memory operations and new authentications. Any new authentications will be performed with all the messages

FFFFFFFFFFFF	A0A1A2A3A4A5	B0B1B2B3B4B5	4D3A99C351DD
1A982C7E459A	000000000000	D3F7D3F7D3F7	AABBCCDDEEFF

Table 4.1: Well known default keys

encrypted without resetting the LFSR state, so information can be extracted from sectors with unknown keys if we previously authenticated in any sector.

4.4 Deployment weaknesses

4.4.1 Deployment with Default Keys

To facilitate testing and integration, the manufacturers ship their chips with default keys. Some of these well known keys are listed in Table 4.1. For different reasons this default keys are in most of the cases not changed.

This kind of weakness is independent of the system itself, if the keys are known, no matter how secure the tag's protocol can be, we will have total access to the data.

4.4.2 Wrong permission configuration

This weakness is similar to the previous one but a little more subtle, it is based on the fact that the chips are distributed with default access permits. A bad configuration of the access permissions can derive in a security breach.

As an example, if the access permissions of one data block are the default we know that it can be read either with key A or B, if any of the keys is a default one the data can be read despite we don't know the other one.

Chapter 5

Tools used for practical attacks

In this chapter we will introduce the different tools used in this project, both hardware and software.

5.1 Hardware

5.1.1 Raspberry Pi

The Raspberry Pi (Figure 5.1) is a credit-card sized computer developed with the intentions of promoting the teaching of basic computer science in schools [10]. It is powered by an ARM processor at 700MHz and 512MB of RAM. It has all the connectors we could find in a standard PC plus some pins for low-level peripherals. Also it can be overclocked without voiding the warranty.

With a power consumption of 3.5W and a Debian based S.O. the Raspberry Pi is a good candidate to substitute a standard PC or laptop and make a standalone device to increase the portability of the RFID reader.

5.1.2 Portable Battery Charger

As a power supply to mount a standalone device we used an external battery pack (see Figure 5.2). The product is described as a power bank to recharge your ipad/iphone on the move, and claims to be able to make the ipad battery last 3.5 extra hours, we can see the specifications in Table 5.1.

This device also has multiple connector adapters that will allow us to recharge the battery

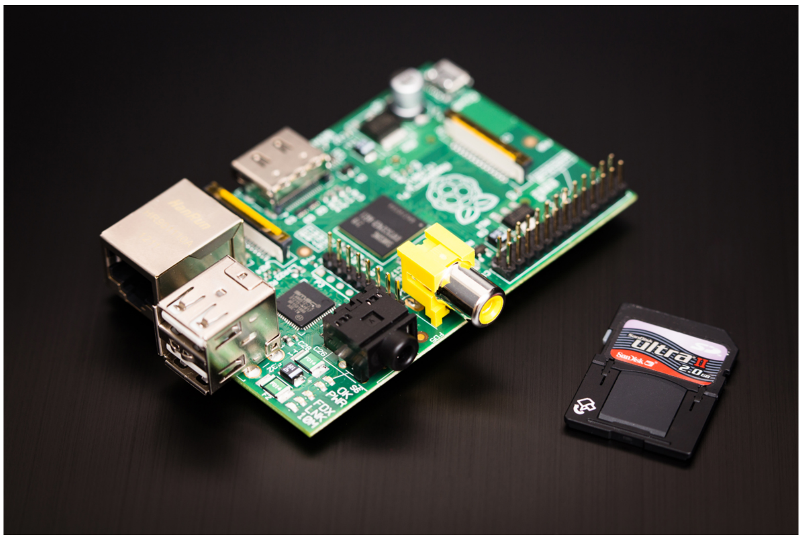


Figure 5.1: Raspberry Pi

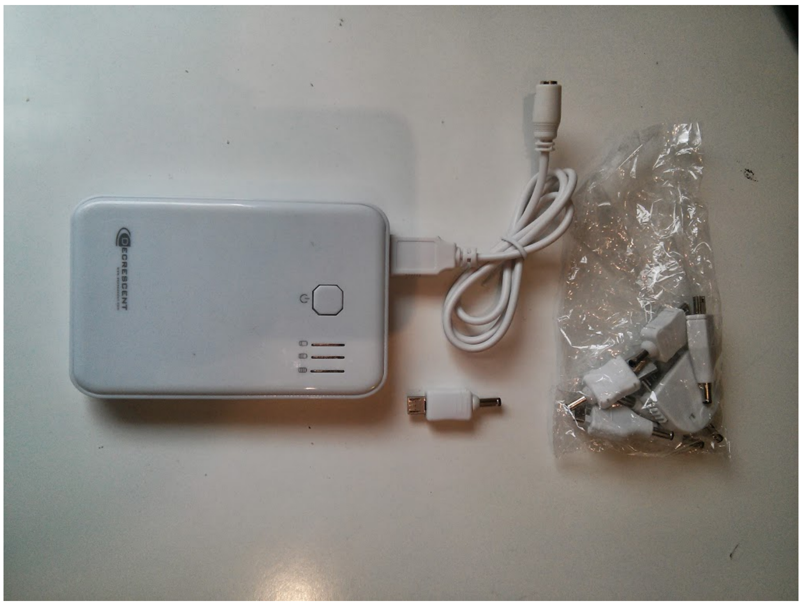


Figure 5.2: Portable battery pack

easily and connect it directly to the Raspberry Pi.

In our experiments we have used it to power the Raspberry Pi so we could use it anywhere. As a result we have seen that it can operate up to 5 hours intensively, we also observed that for the normal usage of our tests it can be used for several days without need to recharge it.

Output capacity	18.5Wh (5000mAh)
USB Output 1	5V, 1A
USB Output 2	5V, 500mA
Charging time	8.5 hours
Input	5V, 1A
Max. output current	1.5A
Dimension	110x70x16mm
Weight	145g

Table 5.1: Battery specs

5.1.3 PN532 NFC v1.3

The PN532 NFC/RFID controller breakout board (Figure 5.3) is a PCD that uses the most popular NFC chip (the PN532 from NXP Semiconductors). It can write and read NFC/RFID Type 1 through 4 tags.

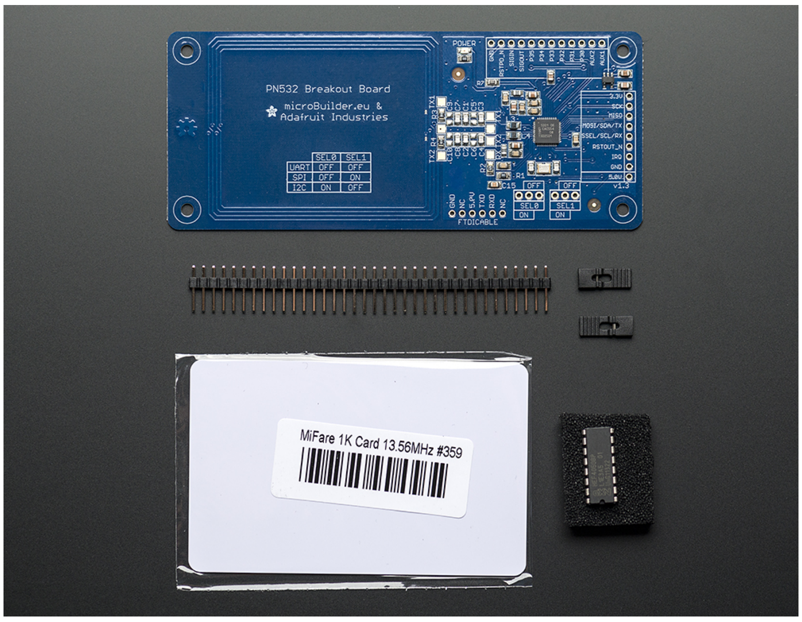

PN532 Breakout Board
microBuilder.eu & Adafruit Industries
MiFare 1K Card 13.56MHz #359

Figure 5.3: PN532 NFC/RFID controller breakout board v1.3

It has tree different ways to communicate with it TTL UART, I2C and SPI, it is also fully supported by libnfc. This allows us to plug in a FTDI cable and use it trough the USB port of any computer.

In this project we will use the TTL UART interface to connect it to the laptop with the FTDI cable and to the Raspberry Pi using either the cable and the GPIO pins as we can see in Figures 5.5 and 5.6.

It also has an integrated antenna designed for using it with NFC enabled systems that makes this device an integrated and cheap solution for experimentation and testing.

As a drawback, the emulation functionality is limited by hardware to only be able to use UIDs starting with `0x08`. These UIDs are used for testing and are no operative in real cases.

5.1.4 MIFARE Classic Customized cards

Reading from the datasheets of the MIFARE Classic chip, we know that the UID is set by the manufacturer and by design can't be altered. Despite this restriction, we have found a Hong Kong provider [14] that sells MIFARE Classic tags with rewritable UID. Using a PCD like PN532 with libnfc [20] and these tags we can make a perfect copy of any tag with known keys.

5.1.5 Smartphones

Android NFC enabled smartphones are good candidates to try the security of the MIFARE Classic tags as there are many free applications that claim to be able to communicate with the tags or exploit some vulnerabilities. For this project we have used a Samsung Galaxy SIII and an LG Nexus 4.

5.2 Software

5.2.1 Operating Systems

Ubuntu 12.04 LTS

Ubuntu is a distribution of the GNU/Linux Operative System. We will use it as a platform for testing the PN532 and libnfc.

Raspbian "Wheezy" 2013-02-09

Raspbian is a free operating system based on Debian, optimized for the Raspberry Pi hardware. This operating system comes with the set of basic programs and utilities that make Raspberry Pi run “out of the box”.

Backtrack 5

Backtrack is a GNU/Linux distribution developed as a penetration testing tool. We observed that it has preinstaled some of the RF tools that we will use on Ubuntu with the PN532.

5.2.2 Programmes and libraries

Libnfc

Libnfc is an open source library written in C to handle Near Field Communication (NFC). The purpose of the libnfc library is to provide developers a way to work with NFC hardware at a higher level of abstraction at no cost. It was released in February 2009 and currently it is at version 1.7. Libnfc can be compiled using Windows Visual Studio in Windows or any C compiler in Linux. The package includes drivers to interact with different kinds of hardware and also comes with some example codes to help us get started.

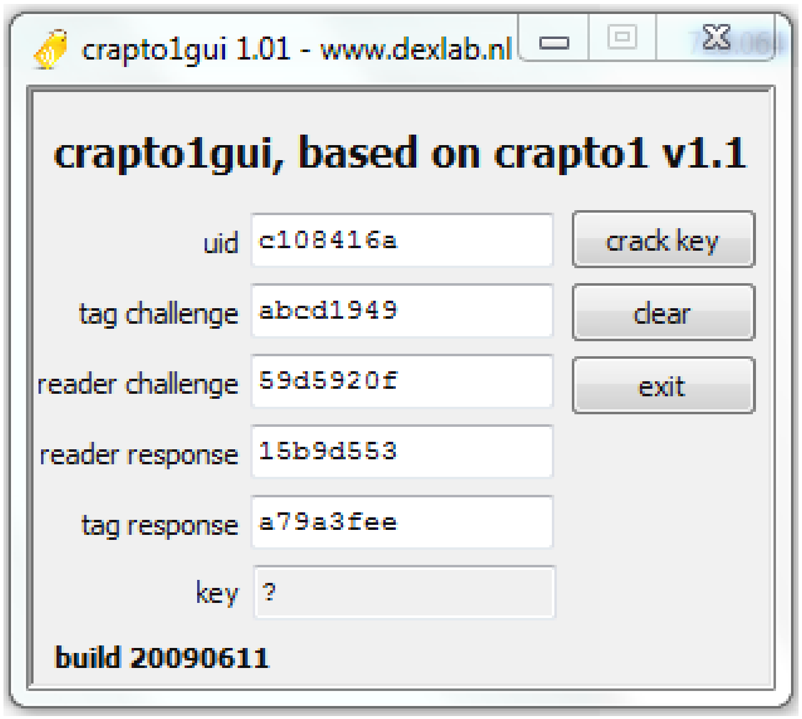
There are some changes in the installation process if we use version 1.7 or older versions, in the older versions the hardware was automatically detected but in the newer ones it has to be specified in which port it is connected as a measure to avoid interferences with other connected hardware and improve the trimmings. Despite this it can be installed in most UNIX like and Windows systems, in the scope of the project this means our laptop, the Raspberry Pi and the PN532 reader.

Crapto1

Crapto1[3] is a C library that implements the Crypto1 cipher in software and also provides some functions supporting the attack from Garcia [11]. It is only a library and does not provide interaction with hardware readers or tags nor any user interface, we can implement one ourselves or (as we found later) we can use some implemented tools that use it [8] to recover the original key from a sniffed authentication (see Figure 5.4). Some Crapto1 functions that we have used to implement the emulation of Crypto1 cipher are described next.

- crypto1_create(key): This command initialises the LFSR with the given key
- crypto1_word: It will return a 32-bit keystream, and updates the LFSR state.
- crypto1_get_lfsr: Returns the LFSR state

- lfsr_recovery64: This function produces the LFSR state after the given 64 keystream bits are generated.
- lfsr_rollback: Roll back the LFSR 32 times to get the previous state. It is an implementation of the attack for the weakness described in Section 4.2.2.


crapto1gui 1.01 - www.dexlab.nl

crapto1gui, based on crapto1 v1.1

uid: c108416a
tag challenge: abcd1949
reader challenge: 59d5920f
reader response: 15b9d553
tag response: a79a3fee
key: ?

build 20090611

Figure 5.4: GUI for Crapto1 tool

MFCUK

MFCUK is an open source C implementation of “Dark Side attack [6] by Courtois” coded by Andrei Costin. It uses Libnfc and Crapto1 library to exploit MIFARE Classic CRYPTO1 weakness. The source code for this library can be downloaded from [5].

MFOC

MFOC is an open source C implementation of “Offline Nested attack [12] by Nijmegan Oakland Group” and coded by Nethemba, an IT security company. It uses Libnfc and Crapto1 library to recover the tag keys using the vulnerabilities seen in 4.1.1, 4.3.1, 4.3.2 and 4.3.3, provided at least one valid KeyA/KeyB of any sector is known, or if the card uses a default key. If a card uses at least one encrypted block with a default key, all the

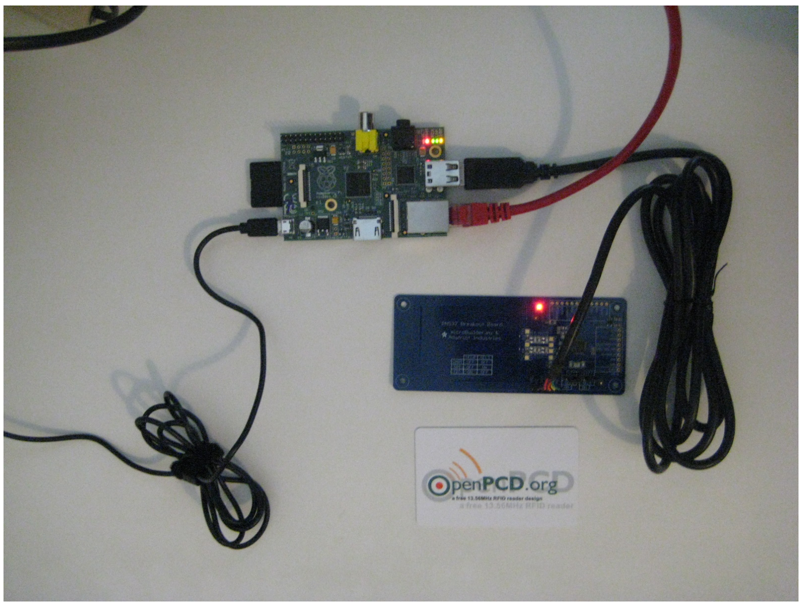


Figure 5.5: PN532 and Raspberry Pi connection with FTDI

other keys can be extracted in minutes. If the card does not use any default keys, one key can be retrieved using the MFCUK application, after which this tool can be used. The source code can be downloaded from [21].

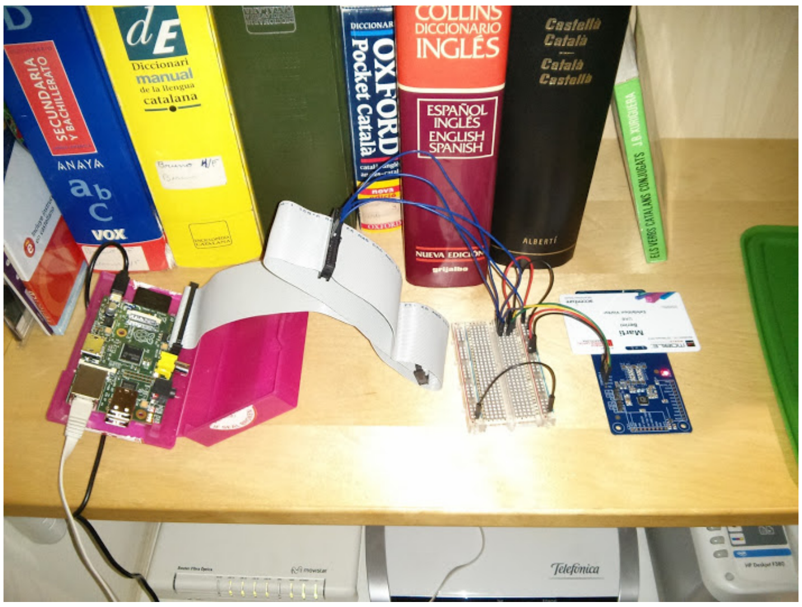
5.3 DIY Standalone device

To increase our mobility during the project we tried to use smartphones to implement our test but we found that they are limited by software so major S.O. changes had to be applied in order to unlock all the features. Furthermore, some smartphone devices have the emulation functionality limited by hardware which eliminates any possible option.

With the same purpose to increase de viability of the attacks by upgrading the mobility of the reader, we have build a standalone device which is capable of acting as a reader and at the same time be portable and have internet connectivity. This device is made from some of the previously presented tools.

First of all, we connected the NFC reader to the Raspberry PI with the FTDI cable as we can see in Figure 5.5. After checking that the PN532 is operative with libnfc installed on the Raspberry Pi we proceeded to make it more mobile by connecting it as described in Adafruit [1] (Figure 5.6(a)), but lately we discovered that the connections could be simplified a lot(which made it even more portable) as we can see in Figure 5.6(b).

Then, to operate it without any screen or keyboard we installed a USB Wifi adapter and


(a) Adafruit connections

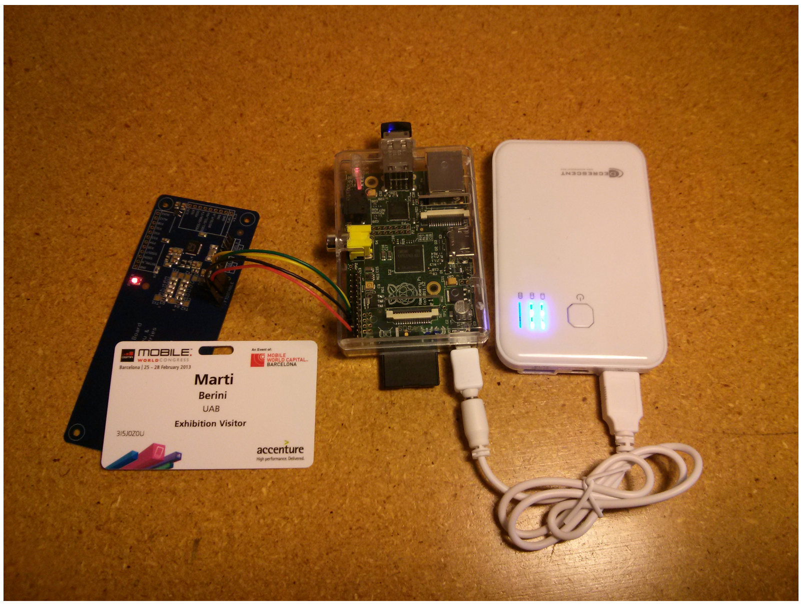

(b) Simplified connections

Figure 5.6: PN532 and Raspberry Pi connections with GPIO pins

configured the ssh server on the Raspberry Pi. Finally we added a battery to recharge ipads as a power supply.

The result (as we can see in Figures 5.6(b) and 5.7) is a standalone device that can work for several hours and maintains a size that allows us to use it everywhere and can be operated from any point in the world.

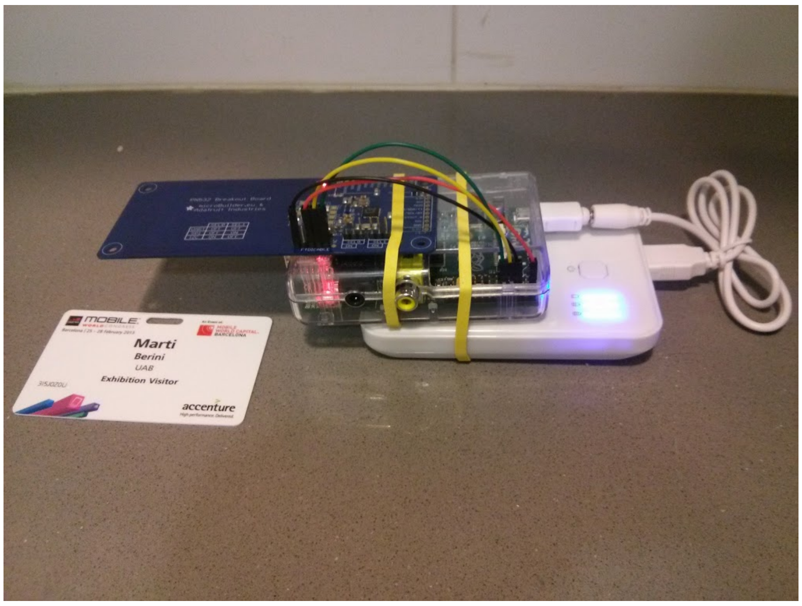


Figure 5.7: DIY standalone device

Chapter 6

Practical feasibility of MIFARE Classic attacks

In this chapter we explain some attacks we have tested, and some real world cases using MIFARE Classic tags. In this last part we will not disclose all the details of the vulnerable systems for its own security.

6.1 Tested Attacks

6.1.1 Key recovery

The objective of this attack is to recover all the keys from a tag to later be able to retrieve the information and modify it. To achieve this goal there are two approaches: Communication interception (as we will see in section **??**) and card only attacks.

Card only attacks are easier to perform as the attacker only needs to be near a tag and act as the reader and can be performed any time while the chances to intercept the communication are lower.

In this category we find two programs that use libnfc and crapto1 to crack the keys: MFCUK and MFOC.

MFCUK

MIFARE Classic Universal toolKit [5] is a toolkit containing samples and various tools based on and around libnfc and crapto1, with emphasis on MIFARE Classic NXP/Philips

# A keys Unknown	# B keys Unknown	# unique Unknown keys	Content retrieval time
11	11	1	1m 38s
11	11	2	2m 21s
11	11	3	3m 7s
0	0	0	0m 5s
1	0	1	0m 39s
2	0	1	10m 1s

Table 6.1: MFOC example times

RFID cards (as seen in Section 5.2.2). It has support for Proxmark3 [19] and PN532 among others.

We used it to perform the "Dark Side Attack" [6]. Basically this attack tries to synchronize with the tag by sending a fixed N_t and receiving encrypted error messages using the vulnerability seen in Section 4.1.1. When it obtains 8 responses using the vulnerability seen in Section 4.3.2, then it tries to reconstruct the possible states using the vulnerabilities from 4.2.1 and 4.2.3 (which gives a list of 2^{16}) and by rolling back we obtain 2^{16} possible keys. Checking the parity bits, it narrows down the possible keys and finally it tries which one is the real key.

MFOC

MFOC is an open source implementation of "offline nested" attack by Nethemba as seen in Section 5.2.2. This attack is much faster than MFCUK but we need to know at least one key of the tag, then it uses the vulnerabilities from Sections 4.3.1 and 4.3.3 to retrieve the keys from the remaining sectors.

By using a combination of the two tools we can perform a complete key recovery by disclosing the first key with MFCUK and the rest with MFOC.

In our experiments MFCUK hasn't worked with the newer tags, as NXP upgraded them some time after the vulnerability was disclosed but with the older ones a key can be recovered within one hour. On the other hand, MFOC worked on most of the cases as expected and in very low times. As we can see in Table 6.1, the time to crack and retrieve the content of a tag is much lower than MFCUK.

6.1.2 Card clonation

The objective of this attack is to replicate a card to fool the genuine reader into thinking that it is communicating with the actual tag. To achieve this we have to retrieve all the data within the PICC.

For a tag with known keys we can retrieve all the information by using the PN532 or an NFC enabled smartphone. In the case of PN532 we used the tool nfc-mfclassic from libnfc, with it we can read or write any data of a genuine tag (except the UID). On the other hand there are some Android Apps, like the one published by NXP, that can retrieve all the information from the tag if the keys are known.

Using any of these tools we could obtain a binary file that is an image of the tag blocks and write it to a new tag, the only difference of these clones to the original tags was the UID as it is written and protected by the manufacturer. As many of the systems using MIFARE Classic authenticate using the UID we needed a way to rewrite it, so we used the custom tags described in Section 5.1.4 to make complete clones. The final result is that the standard readers can't distinguish between the original tags and the clones.

6.1.3 Card Emulation

The objective of this attack is to use some hardware to fool the reader into thinking that it is communicating with an authentic tag but without using a real MIFARE Classic chip. These kinds of attacks have more flexibility than the Clonation Attack because the data within the emulated tag can be modified easily.

To perform these attacks there are two main approaches:

- NFC enabled smartphones
- Dedicated hardware

Due to the time constraints of the Project we focused on the dedicated hardware approach using the PN532.

Dedicated hardware

The PN532 has the emulation features limited by hardware to only being able to use test UIDs (first byte equal to `0x08`), but by disabling the chip communication handlings after a fake initialization to send raw data we can forge messages and send any command, even real UIDs.

We developed a program to fully emulate a MIFARE Classic tag and its communications. By using libnfc, crapto1 and our DIY standalone device we want to use any UID and emulate the content of any tag (Figure 6.1(a)).

The main problems we found were that there was only detailed information about simple reader behaviours, that the implementation of the procedure is not exactly as the theoretical definitions we found, the hardware restriction to emulate UIDs and the access to a reader for the tests.

With this program we have been able to replicate the complete process and fool some readers with loose timeouts that not only read the UID but also authenticate by reading some of the contents of the tag. Despite that, most of the readers have very restrictive timeouts, so the use of the PN532 trough USB is discarded and the DIY standalone device doesn't work in all the situations.

The multi sector nested authentication has not been implemented yet, we used a simple authentication for the tests.

6.2 Real World RFID deployments

In this section we describe some of the vulnerabilities found in real systems that use the MIFARE Classic tags as RFID device.

6.2.1 Mobile World Congress 2013

The first case we present is the Barcelona Mobile World Congress 2013. This year the main novelty was, as they called it, the NFC experience. So the ticketing was made with MIFARE Classic tags, also, inside the MWC, every company's stand had readers to participate in contests and control visits. There was also an application for NFC smartphones that emulated the Access badge.

Using our badge we found that it was a MIFARE Classic 1k tag. On the first test of the tag we found that only the 12 first sectors used non-default keys, so we could use MFOC to retrieve all the information in less than 2 minutes. Comparing it to other badges we found that the same keys are used in every tag, this makes it possible to retrieve all the information of any tag within 5 seconds with our DIY standalone device.

Then, we proceeded to identify the tag's retrieved data and we found that the first 2 sectors contained the visitor registration numbers. The subsequent sectors contain the personal data of the user in ASCII such as names, email, phone number, ID (DNI) and postal code; the final sectors contain padding data too.

As we didn't find any data that determines access restrictions we think that it is stored in the back-end of the system. Then the most feasible attack is impersonating other attendants as well as stealing their private information, since such information is included in the card.

Finally we tried the NFC smartphone application whose only purpose was to emulate the badge while showing the user details on the phone screen. While using the LG Nexus 4 we observed that the NFC was a little bugous, so in some cases we were allowed to enter the facilities by showing the screen of our phone. Although this is not a problem of the RFID tags it is an important security breach for the organization.

6.2.2 Building and facility access controls

The test for the access control applications has been performed in three different facilities. In the first test case we observed that the access badges to access the building authenticate only by using the tag's UID, this makes the system totally insecure as the UID can be read instantly from any tag and used later using a card clone or by emulation (a brute force attack is also feasible as the UID has only 4 bytes).

In the second test case we found that the tags had some personal information to authenticate together with the UID, this practice makes the brute force attack harder and would make the information extraction slower except for the fact that the wrong configuration of the tag permissions allowed us to access the data with the default B keys (they configured the A keys only).

In the third test case we observed that the access permissions to the different zones of the building were codified in the tag instead of performing a personal authentication and granting access by identity. This configuration makes it possible to modify the tag information and gain access to any zone and it makes it harder to differentiate between the users.

6.3 Vending and renting applications

We have seen that some self-service cafés use tags to accelerate the payment process. For security reasons they store the balance of the users on a server instead of the tags. However, to identify the users the system only reads the UID of the tags, as the ticketing machine is unsupervised, anyone can try UIDs until a valid one is found and spend all the credit.

A similar problem applies to a well known renting service, in this case the problem is not the credit but using the service impersonating another customer as this service only uses the UID of the tags to authenticate.

In another case we, observed that in some buildings personal ID badges are used by all the working personnel. For security reasons they use the badges as a measure to control the access to certain areas of the building, each worker only has access to the facilities related to his or her job and position. These tags are also used as a wallet for the vending machines.

We found that these badges use the MIFARE Classic 4k chip, so we started the testing. The first relevant result was that most of the sectors are blank and protected with the default key and permissions, this allowed us to perform a card only attack using MFOC to retrieve all the tag's information.

Analysing the content we observed that there were only four unique non-default keys (two A and two B) and three kinds of permissions. The data contained in the tag can be separated in two main categories from its sector positions.

1. Data stored in sectors 4 through 8 which has one A key, one B key and two kinds of permissions.
2. Data stored in sectors 38 through 40 which has one A key and one B key.

Testing the two categories we found that category one is the one used as a wallet and category two is for access control. Due to the restricted access to badges for testing we will focus on the wallet data.

By reading the tag's memory content with different amounts of money we found that some of the sectors were modified, this made us think that the money is stored in the tag instead of a back-end system.

In our next test we demonstrated that it is possible to spend any number of times the same money by keeping a backup of the charged tag, this also confirmed our assumption that the money is stored in the tag.

Then we focused on the analysis of the wallet data to disclose how it stores the user's money, our first observations from a tag with different amounts of money can be seen in Table 6.2.

After the previous observations we can discard the static memory sectors and proceed with less data to carry out our analysis; the remaining sectors (8, 6 and 5) can be also separated in two groups according to their access permissions As we can see in Tables 6.3 and 6.4, comparing this data to the access conditions (Tables 3.2 and 3.3) we observed that sector 5 is used as a value block and sectors 6 and 8 are used as data blocks.

Sector	Modified	Comments
4	false	Deployer Company
5	true	Different permissions than sectors 4, 6,7 and 8
6	true	-
7	false	-
8	true	-

Table 6.2: Wallet initial data from multiple readings

	Hex	BIN		C1	C2	C3
Byte 6	60	0110 0000	Block 0	1	1	0
Byte 7	F7	1111 0111	Block 1	1	0	0
Byte 8	89	1000 1001	Block 2	1	0	0
Byte 9	00	0000 0000	Block 3	1	1	1

Table 6.3: Restricted building tag, Sector 5 permissions

Testing these data blocks we discovered that sectors 6 and 8 are used to store the recent usage history of the tag. Comparing all the data, we extended the information as follows:

- Sector 8 Block 0: In this block the last amount of money charged into the tag is stored with a check code.
- Sector 8 Block 1: In this block it is stored penultimate amount of money charged into the tag with a check code.
- Sector 6 Block 0: In this block the code of the last machine used (either for replenishing or for buying) is stored.

Finally we analysed the value block (sector 5 block 0), as described in the MIFARE Classic documentation [22]. It contains a value stored once inverted and twice non inverted. Extracting the values of the tag with different amounts of credit and swapping the bytes we obtained the results of Table 6.5.

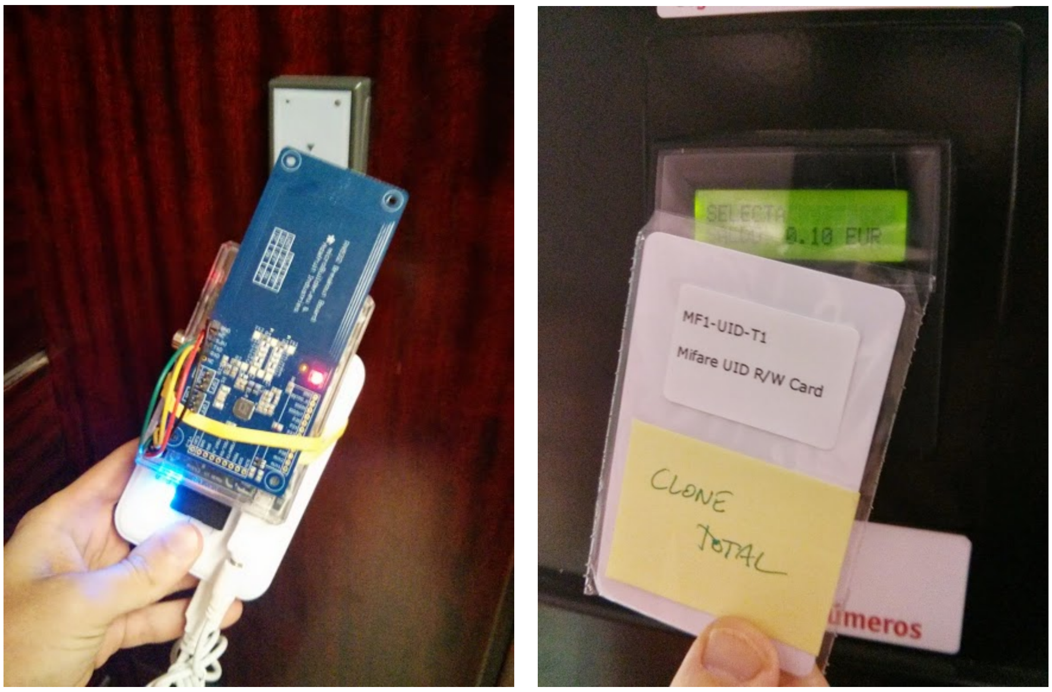
These results mean that the credit inside the tag is not protected so any amount can be written in the badges without paying for it as we can see in Figure 6.1(b).

	Hex	BIN		C1	C2	C3
Byte 6	70	0111 0000	Block 0	1	0	0
Byte 7	F7	1111 0111	Block 1	1	0	0
Byte 8	88	1000 1000	Block 2	1	0	0
Byte 9	00	0000 0000	Block 3	1	1	1

Table 6.4: Restricted building tag, Sectors 6 and 8 permissions

Credit	HEX Value	Swapped HEX Value	DEC Value
0.70€	BC 02 00 00	02 BC	700
0.80€	20 03 00 00	03 20	800
0.90€	84 03 00 00	03 84	900
0.25€	FA 00 00 00	FA	250
0.10€	64 00 00 00	64	100

Table 6.5: Restricted building tag, credit values



(a) DIY standalone device in use (b) Wallet credit manipulation

Figure 6.1: MIFARE Classic attacks

Chapter 7

Conclusions

With the obtained results we can conclude that the MIFARE Classic tags are very insecure, despide that a high number of enterprises still use them by the lack of knowledge or to save the cost of updating the systems with a newer technology, but using a better configuration would increase the security of these systems keeping a low cost.

In this chapter we summarize the principal problems of the MIFARE Classic tags and our recommendations to minimize the risk.

7.1 MIFARE Classic Problems

The first and most common vulnerability in the implementation of the different systems is in the default keys, we haven't found any tag without at least one default key. This means that all the systems that we have analysed can be exploited by the card only attack MFOC.

The following list summarizes the main problems of MIFARE Classic tags found in this project:

- MIFARE Classic tags are very insecure.
- MIFARE Classic tags are widely used.
- Some systems use only the UID for authentication, and UIDs can be emulated easily.
- Some systems have a wrong permissions configuration, *i.e.* they set the A key but the permissions allow to use the default B key to access the data.
- The data stored in the tags is usually in plain text.

- The privileges that grant each tag are based on the information inside the chip instead of carrying out a good personal authentication.
- It seems very secure for the general public as they don't know what it is or how it works

7.2 Implementation best practices

The principal recommendation is, if possible, avoid using MIFARE Classic tags; if that is not an option the following list summarizes the best practices for the usage of these tags.

- Don't leave any default key (neither A or B) in any sector
- Never use only the UID for authentication, the cryptographic features of the tags must be used.
- Use the most restrictive permissions possible for the use we are going to do of that tag.
- Cipher the data and use check codes in different sectors to detect manipulation.
- Include history checks or use tokens to avoid replay of operations.
- If used as a wallet don't store the credit in the tag, use a back-end system to avoid credit manipulation.

7.3 Future Work

During the execution of this project some aspects have been found that may be of interest for future investigations.

- Test if a relay attack or a communication interception is feasible.
- Study how much the range of these tags can be upgraded to perform attacks or intercept communications (Figure 7.1).
- Study MIFARE Plus: According to the NXP documentation it is an evolution of MIFARE Classic and it uses Crypto1+AES, and it is used by some banks as a wallet.

- Credit cards with NFC as Visa (paywave) and MasterCard (paypass) don't use any kind of encryption in the communications and all the information can be retrieved without using any key. As their usage is even wider than the MIFARE Classic tags it may be an important case of study.
- Smartphones with NFC are used as a substitute of the NFC credit cards. So they inherit the same problems as the credit cards as well as the risks from a high connectivity and multitasking.
- Test if NFC enabled smartphones can emulate MIFARE Classic tags or perform a relay attack using Cyanogen Mod [15] and a dedicate application [16].
- PN532 is simpler and cheaper than a proxmark3 but it is limited by the response time, so adapting the emulation software to the proxmark3 would increase the performance.

Figure 7.1: Someone sniffing RF communications at MWC

Bibliography

[1] Adafruit. Nfc on raspberry pi. http://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi/overview. Accessed March 24, 2013.

[2] Corey Benninger and Max Sobell. Nfc for free rides and rooms on your phone. http://media.risky.biz/EUSecWest-SoBenn-Transit2012-Preview.pdf. Accessed Octover 19, 2012.

[3] blapost. Crapto1. http://code.google.com/p/crapto1. Accessed February 25, 2013.

[4] M. Bolic, D. Simplot-Ryl, and I. Stojmenovic. *RFID systems: research trends and challenges*. Wiley, 2010.

[5] Andrei Costin. Mfcuk mifare classic universal toolkit. http://code.google.com/p/mfcuk/. Accessed February 27, 2013.

[6] Nicolas T Courtois. The dark side of security by obscurity. *International Conference on Security and Cryptography*, 2009.

[7] G. de Koning Gans, J.H. Hoepman, and F. Garcia. A practical attack on the mifare classic. *Smart Card Research and Advanced Applications*, pages 267–282, 2008.

[8] dexlab. Crapto1gui. http://www.dexlab.nl/downloads.html#crapto1gui. Accessed March 28, 2013.

[9] K. Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. Wiley, 2010.

[10] The Raspberry Pi Foundation. Raspberry pi. http://www.raspberrypi.org/. Accessed January 15, 2013.

[11] Flavio Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter Van Rossum, Roel Verdult, Ronny Schreur, and Bart Jacobs. Dismantling mifare classic. *Computer Security-ESORICS 2008*, pages 97–114, 2008.

[12] Flavio D Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly pickpocketing a mifare classic card. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 3–15. IEEE, 2009.

[13] Mads Haahr. Random.org. `http://www.random.org`. Accessed March 2, 2013.

[14] Wah Lok Ind. rfidshop. `http://www.rfidshop.com.hk/`. Accessed February 10, 2013.

[15] Steve Kondik. Cyanogenmod. `http://www.cyanogenmod.org/`. Accessed April 15, 2013.

[16] Eddie Lee. Nfc hacking: The easy way. `http://blackwinghq.com/assets/labs/presentations/EddieLeeDefcon20.pdf`. Accessed April 10, 2013.

[17] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-engineering a cryptographic rfid tag. In *Proceedings of the 17th conference on Security symposium*, pages 185–193, 2008.

[18] Karsten Nohl and Henryk Plötz. Mifare, little security, despite obscurity. In *Presentation on the 24th congress of the Chaos Computer Club in Berlin*, 2007.

[19] Proxmark3. Proxmark 3. `http://proxmark3.com/`. Accessed April 13, 2013.

[20] Roel. libnfc. `http://code.google.com/p/libnfc/`. Accessed February 23, 2013.

[21] romuald. Mfoc mifare classic offline cracker. `https://code.google.com/p/mfoc/`. Accessed March 7, 2013.

[22] NXP Semiconductors. Mifare classic. `http://www.nxp.com/`. Accessed October 16, 2012.

[23] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[24] W.H. Tan. Practical attacks on the mifare classic. *Imperial College London*, 2009.

Firmat: Marti Berini Sarrias

Bellaterra, juny de 2013

Resum

Aquest projecte inclou una aproximació als conceptes de RFID i targetes contactless centrant-se en l'ampliament usat MIFARE Classic chip. L'objectiu principal es mostrar el seu funcionament i les seves vulnerabilitats, així com alguns exemples pràctics fent una anàlisi de diferents serveis que les utilitzen.

Resumen

Este proyecto incluye una aproximación a los conceptos de RFID y tarjetas contactless centrándose en el ampliamente usado MIFARE Classic chip. El objetivo principal es mostrar su funcionamiento y sus vulnerabilidades, así como algunos ejemplos prácticos haciendo un análisis de diferentes servicios que las utilizan.

Abstract

This project includes an introduction to the concepts of RFID and contactless cards by focusing on the widely used MIFARE Classic chip. The main objective is to show how it works and its vulnerabilities, as well as some practical examples making an analysis of different services that use it.