Ciphertext-only cryptanalysis on hardened mifare classic cards extended

Meijer, C.F.J.

Award date:
2016

EINDHOVEN UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Ciphertext-only Cryptanalysis on Hardened Mifare Classic Cards Extended

The paradox of keeping things secret

Author:
Carlo MEIJER

Supervisors:
dr. ing. Roel VERDULT
prof. dr. Boris SKORIĆ

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in

Information Security Technology
Department of Mathematics and Computer Science

August 17, 2016

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Computer Science
Department of Mathematics and Computer Science

Master of Science

Ciphertext-only Cryptanalysis on Hardened Mifare Classic Cards Extended
The paradox of keeping things secret

by Carlo MEIJER

Despite a series of attacks, MIFARE Classic is still the world's most widely deployed contactless smartcard on the market. The Classic uses a proprietary stream cipher CRYPTO1 to provide confidentiality and mutual authentication between card and reader. However, once the cipher was reverse engineered, many serious vulnerabilities surfaced. A number of passive and active attacks were proposed that exploit these vulnerabilities. The most severe key recovery attacks only require wireless interaction with a card. System integrators consider such card-only attacks as one of the most serious threat vectors to their MIFARE Classic-based systems, since it allows the adversary to avoid camera detection, which is often present at an access control entrance or public transport gate. However, all card-only attacks proposed in the literature depend on implementation mistakes which can easily be mitigated without breaking backwards compatibility with the existing reader infrastructure.

Consequently, many manufactures and system integrators started to deploy "fixed" MIFARE Classic cards which are resilient to such vulnerabilities. However, these countermeasures are rather palliating and inadequate for a cryptographically insecure cipher such as CRYPTO1. In support of this proposition, we present a novel cipher-text card-only attack that exploits a crucial and mandatory step in the authentication protocol and which solely depends on the cryptographic weaknesses of the CRYPTO1 cipher. Hence, in order to avoid this attack, all cards and readers should be upgraded to support an alternative authentication protocol which inherently breaks their backwards compatibility. Our attack requires only a few minutes of wireless interaction with the card, in an uncontrolled environment and can be performed with consumer-grade hardware. The information obtained allows an adversary to drop the computational complexity from 2^{48} to approximately 2^{30}, which enabled us to practically recover a secret key from a hardened MIFARE Classic card in about 5 minutes on an single core consumer laptop.

Acknowledgements

This reasearch is conducted as a master thesis for the master Information Security Technology, which is offered by Eindhoven University of Technology. Though, the entire research took place at the Radboud University Nijmegen.

This thesis is an extension of the paper [MV15], by myself and Roel Verdult. This thesis contains, among other extensions, proofs of lemmas that were omitted in the original paper due to space restrictions. Furthermore, a number of errors in the paper have been identified and corrected in this thesis.

I would like to take the opportunity to thank my supervisor, Roel Verdult. I still remember my first appearance in Roel's office, exchanging ideas about what an interesting research topic. Although I really liked the idea of performing cryptanalysis on hardened MIFARE Classic, I was reluctant since, at that time, I was neither experienced nor skilled in the field of cryptanalysis. However, Roel pointed me to some literature and convinced me to give it a try. Roel's confidence in my abilities, useful insights and guidance have been crucial ingredients during this research, as well as for my personal development. It has been a great pleasure to work with you, Roel.

The next person I would like to thank is Tanja Lange. She was my intended supervisor and has supervised me throughout the research. Unfortunately, due to time constraints, I had to switch to Boris Skorić during the time when the graduation took place. Although we didn't see each other much in person during the research, her feedback has contributed immensely to the quality of my paper and this thesis. As for her contribution: I remember the submission deadline for my paper being 1PM, and receiving the last batch of feedback at 6.39AM, so I still had enough time to merge them in the final revision. I think that says it all. Thank you so much.

Finally, I would like to thank Boris Skorić, who helped me a lot getting all the paperwork done for my graduation. I first introduced myself about a month before my defense. Thank you so much for your support and the nice and speedy arrangement.

The responsible disclosure, giving a talk at a scientific conference, the media attention. Working on this research has been a truly unforgettable experience. Thanks everyone involved for making it possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

MIFARE Classic cards occupy a considerable part of the contactless smartcard market. Such cards offer, in addition to simple identification, a modest amount of memory and cryptographic capability, making them suitable for applications such as access control and fare collection systems.

The MIFARE Classic cards are still widely deployed in many public transport payment systems. Examples of such systems include the Oyster Card[1] in London, the Charlie Card in Boston[2], the EasyCard in Taiwan[3], and the OV-chipkaart in the Netherlands[4]. In addition to public transport, it is also still used for contactless access control systems, integrated in many office buildings, including even high-security facilities such as ministries, banks, prisons and nuclear power stations.

MIFARE Classic cards are compatible with parts 1 to 3 of the ISO/IEC 14443-A standard [ISO01], specifying physical characteristics, radio frequency interface and the anti-collision protocol. However, part 4 of the standard, describing the transmission protocol, is not implemented. Instead, it uses its own secure communication layer. In this layer, MIFARE Classic cards communicate encrypted by using a proprietary stream cipher, named CRYPTO1, to provide data confidentiality and mutual authentication between card and reader. The manufacturer, NXP Semiconductors, never publicly released details of the cipher, nor the communication layer used in MIFARE Classic. However, both have been successfully reverse engineered and their workings are published in the literature [NESP08, GKGM+08].

Shortly after reverse engineering, serious vulnerabilities were identified and several attacks surfaced that demonstrated to be mountable in practice. For instance, the attacks described in [GKGM+08] and [KGHG08] enable an attacker to recover encryption keys used during communication and manipulate the related data segments on the card. This allows the attacker to (partially) clone the card. However, all the attacks mentioned above require the attacker to have access to a legitimate reader, or eavesdrop on genuine communication. Not long after, several attacks were proposed [GRVS09, Cou09, CHC+14] which demonstrate how to recover all encryption keys, and hence clone a full card, only by wirelessly interacting with it. These card-only attacks are the most effective way for an attacker to compromise the security of a MIFARE Classic based system. Since a perpetrator can perform such an attack in an isolated, non-controlled environment, they cause the maximal amount of damage against the least possibility of detection.

[1] http://oyster.tfl.gov.uk
[2] http://www.mbta.com/fares_and_passes/charlie
[3] http://www.easycard.com.tw
[4] http://wwww.ov-chipkaart.nl

Interestingly, all card-only attacks proposed in the literature rely on non-cryptographically related implementation flaws. Moreover, these implementation issues can be mitigated by issuing replacement cards where such flaws are removed, without even breaking backwards compatibility to the original MIFARE Classic cards. However, these replacement cards do not provide a solution to the actual problem, which is the insecurity of the underlying cryptographic algorithm and authentication protocol.

Table 1.0.1 gives a (non-exhaustive) overview of MIFARE Classic compatible cards, together with revisions made to the original MIFARE Classic card with respect to security.

TABLE 1.0.1: The MIFARE Classic compatible cards

Card	*a*	*b*
MIFARE Classic	×	×
MIFARE Classic EV1	✓	✓
MIFARE Plus in security level 1	✓	✓
MIFARE SmartMX in Classic mode	✓	✓
NXP manufactured NFC controllers with MIFARE Classic card emulation	✓	✓
Third party licensed products, such as the Infineon SLE-66	✓	✓
Unlicensed MIFARE Classic clones, such as the Fudan FM11RF08	×	×
Newer clones, such as the one used in Taiwan EasyCard 2.0	✓	×

[a] Has a proper pseudo-random number generator
[b] Does not send encrypted error code after failed authentication

1.1 Research Question

Are the modifications made to drop-in replacement MIFARE Classic cards sufficient to prevent practical card-only attacks?

Previous research has shown the security properties of the CRYPTO1 cipher used in the MIFARE Classic protocol are much weaker than advertised [GKGM+08, GRVS09, Cou09]. Most notable are the card-only attacks, which enable an attacker to recover secret keys from a card in an uncontrolled environment, thus avoiding camera detection, using only consumer-grade hardware. Many system integrators felt the need to respond in order to counter widespread abuse. Some system integrators decided to migrate their MIFARE Classic-based infrastructure to more secure alternatives. Others decided to deploy replacement cards with the implementation flaws removed, thus mitigating all card-only attacks known at the start of this research, while remaining backwards compatible with the reader infrastructure. Still, a number of very efficient key recovery attacks requiring reader communication remain unaffected.

To our judgment, this approach is inadequate for a fundamentally weak cipher such as CRYPTO1. Therefore, this research focuses on building a novel card-only attack on cryptographic design flaws that cannot be removed without breaking backwards compatibility.

1.2 Contribution

In this thesis we propose a ciphertext-only attack against MIFARE Classic cards, which only requires wireless interaction with the card for a few minutes with consumer-grade

hardware. We have fully implemented and tested our attack in practice and recovered secret keys within minutes from various hardened MIFARE Classic cards. In order to give a better estimate of the average running-time, theoretical boundaries and a performance analysis based on simulations are given in this thesis.

The attack proposed in this thesis requires the attacker to know at least one single key in advance for each card that she wants to recover keys from. In practice, however, this is typically the case. The first key can be retrieved by eavesdropping only one genuine authentication or two failed authentication attempts, see [VKGG12, GKGV12, GKGM+08] for more details. However, in many situations, this is not necessary since most deployed systems leave default keys, which are set during manufacturing, intact for unused sectors. Additionally, nearly all deployed systems that use key diversification leave at least one sector key non-diversified, namely for storing the diversification information. Moreover, the manufacturer guidelines for system integrators [MAD07] especially recommends setting up the key diversification in such a way.

1.3 Overview

This thesis is organized as follows. Chapter 2 gives a broad overview of the literature related to attack stream ciphers in general and the more closely related papers that attack the MIFARE Classic cryptosystem. Chapter 3 introduces the memory layout, cipher description and authentication protocol that is used by a MIFARE Classic card. Vulnerabilities to the MIFARE Classic cryptosystem are addressed in Chapter 4. Then, a novel method to attack the cipher is proposed in Chapter 5, which is followed by Chapter 6 where we analyze the performance of the attack. Finally, a summary of the attack and its practical implications are given in Chapter 7.

Chapter 2

Related work

In this section we first explore generic attack techniques and then highlight the different methods that were proposed in the literature to attack a MIFARE Classic card. For each previously proposed attack we analyze its significance and the corresponding practical implications.

2.1 General Stream Cipher attacks

In the last decades, three main techniques were introduced to attack LFSR-based stream ciphers, such as the one used in a MIFARE Classic card. First, the guess-and-determine attack can be mounted if the cipher does not use its complete internal state to compute a keystream bit, despite several well-known historical recommendations in the literature [Kuh88, And91, Gol96]. Besides the MIFARE Classic cryptosystem, many proprietary LFSR-based stream ciphers [Gol97, DHW+12, VGB12] lack this property and are therefore vulnerable to partial and incremental internal state guessing. Secondly, there is the correlation attack, which was originally proposed by Siegenthaler [Sie84, Sie85] and later improved by others [MS88, And95, CS91, CCCS92, JJ00, CJM02]. It exploits the weakness of a relation between the internal state bits and the keystream. The filter function of the MIFARE Classic cryptosystem uses some input bits which are more influencing than others. This allows for a similar correlation attack to be mounted. Finally, more recently, various algebraic attacks on general stream ciphers were proposed [CP02, AK03, FJ03, CM03]. A property of a linear Boolean function is the possibility to postpone an evaluation. Computational problems can be written as a system of Boolean equations [TT80], which are formalized during a cryptographic attack. Instead of computing the outcome directly, a combination of these equations can be solved by well-known techniques such as Gaussian elimination [Hil29, Mul56, Mar57, Str69]. Because of the regularity of the chosen indexes of the filter function inputs, the MIFARE Classic cryptosystem is particularly vulnerable to these attacks.

2.2 Attacks on MIFARE Classic

2.2.1 Keystream recovery attack

The first practical attack against MIFARE Classic was carried out in 2008 by de Koning Gans, Hoepman and Garcia [KGHG08]. It recovers the keystream used in a transaction between a reader and a card. Due to a weak pseudo-random number generator (PRNG), the resulting keystream can be kept constant and reused by a malleability attack. For such an attack knowledge about the secret key and encryption algorithm is not required.

2.2.2 Genuine reader attacks

The inner workings of CRYPTO1 were reverse engineered shortly after the first attack. The Linear Feedback Shift Register (LFSR) is available in [NESP08] and the non-linear filter function and authentication protocol are shown in [GKGM+08]. The latter proposed a serious attack that exploited weaknesses in the filter function, allowing an attacker to invert the filter function in an extremely efficient way. This enables an attacker to recover the secret key from a single captured authentication session within a fraction of a second on ordinary hardware.

2.2.3 Card-only attacks

A number of attacks that require only interaction with a card were introduced by Garcia, van Rossum, Verdult and Wichers Schreur in [GRVS09]. The first can be mounted against a single authentication and requires precomputation tables. The knowledge of one sector key allows for their second attack, which is mounted against a nested authentication, it is extremely fast, and does not require precomputation tables. The attack against a single authentication was improved by Courtois [Cou09]. This attack does not require any precomputation and is faster than the one proposed by Garcia et al.

More recently, Chiu, Hong, Chou, Ding, Yang and Cheng [CHC+14] proposed a card-only attack that does not depend on a weak random number generator, instead it exploits another implementation mistake, the encrypted error code response. Unfortunately, this attack requires a large amount of online generated traces which significantly increases the total running time of their attack.

In fact, all card-only attacks that are proposed in the literature depend on either a weak random number generator, or keystream leakage through known error messages, or both. A straightforward solution against these attacks is to replace the vulnerable cards with modified cards that do not contain any known implementation mistake, but which are still fully compatibility with the MIFARE Classic protocol. In this thesis we will refer to such modified cards as *hardened* MIFARE Classic cards, examples of such cards are given in Table 1.0.1.

Chapter 3

Background

MIFARE Classic cards are one of the first generation RFID tags designed in the 90s. During that time, it was a common practice by the industry to design proprietary RFID products. Such a proprietary design often contains custom defined modulation/encoding schemes, packets, checksums, instruction sets and in some cases even custom made cryptographic algorithms and authentication protocols.

There is not much wrong in designing a custom way of RFID communication. It allows the industry to optimize products and boost the performance for specific applications. However, this argument certainly does not hold for the proprietary and secret cryptosystems. Designing secure algorithms is proven to be a difficult task without feedback from the scientific community [Ker83, JS97, SN97, Ver15].

3.1 MIFARE Classic cards

The MIFARE Classic card is an interesting example which implements besides a custom communication protocol, a proprietary cipher and also an authentication protocol. The datasheet [PHI98] of the MIFARE Classic card suggests that the security properties are compliant to standardized authentication protocols [ISO99]. However, in practice the security properties are significantly weaker than advertised.

3.2 Memory Structure

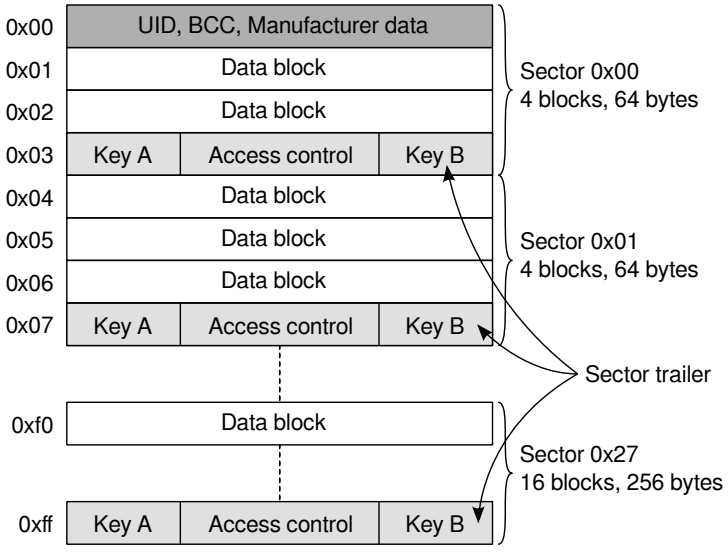


FIGURE 3.2.1: Memory layout of a MIFARE Classic card

A MIFARE Classic card is essentially a memory chip with wireless communication and encryption capabilities. The memory is divided into sectors, each of which is further divided into blocks of sixteen bytes each. The last block of each sector is the sector trailer and stores two secret keys and the access conditions for that sector. The MIFARE Classic 1k has 16 sectors consisting of 4 data blocks each. The MIFARE Classic 4k has 40 sectors, where the first 32 sectors consist of 4 data blocks and the remaining 8 sectors consist of 16. A schematic overview of the memory layout is shown in Figure 3.2.1.

Note that block 0 of sector 0 is different in the sense that its contents have a special meaning. The first 4 bytes contain the card's unique identifier (uid), followed by its 1-byte *bit count check* (bcc). The bit count check is computed by bitwise XORing the uid bytes together. The remaining bytes contain data set by the manufacturer. The data in this block is write-locked during the manufacturing process and hence cannot be modified. However, a notable exception exists for unofficial uid-changeable MIFARE Classic clones.

To perform an action on a specific block, the reader must first authenticate itself for the sector containing that block. The access conditions determine, for both keys separately, which actions are allowed to be performed by the reader.

3.3 Notation

The mathematical symbols are defined as follows:

Let $\mathbb{F}_2 = \{0, 1\}$ be the field of two elements (or the set of Booleans). The vector space $\mathbb{F}_2^n$ represents a bitstring of length n. Given two bitstrings x and y, xy denotes their concatenation.

Given a bitstring $x \in \mathbb{F}_2^n$, x_i denotes the i-th bit of x, where $0 \leq i < n$. Furthermore, $x_{[i,j]}$ denotes the substring of x that starts at index i and ends at index j, inclusive. Thus, representing the substring $x_i x_{i+1} \dots x_j$. For instance, given the bitstring $x = \texttt{0x010203} \in \mathbb{F}_2^{24}$, then byte $x_{[16,23]} = \texttt{0x03}$ and the bits $x_{22} = x_{23} = 1$.

The symbol ϵ represents the empty bitstring, $\oplus$ denotes the bitwise exclusive-or (XOR) and $\overline{x}$ denotes the bitwise complement of x. The large XOR symbol denotes an inner XOR of multiple bits, i.e. a repeated logical exclusive-or. For example $\bigoplus_{i=0}^{3} x_i = x_0 \oplus x_1 \oplus x_2 \oplus x_3$.

3.4 Cipher and Tag Nonces

The cipher is based around a *Linear Feedback Shift Register (LFSR)*, which, on every step, shifts its contents one position to the left. The most significant bit is then discarded and a new least significant bit is generated by applying the feedback function and, during the authentication phase, feeding the input. Besides being shifted, all other bits remain unaffected.

To increase readability, we adapt the same notation as used in [GRVS09] and comply with their formalization. Concretely, the cipher feedback, filter and tag-nonce related functions of CRYPTO1 are specified in Definition 3.4.1–3.4.4.

Definition 3.4.1. The cipher feedback function $L\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by $L(x_0 x_1 \dots x_{47}) := x_0 \oplus x_5 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{19} \oplus x_{24} \oplus x_{25} \oplus x_{27} \oplus x_{29} \oplus x_{35} \oplus x_{39} \oplus x_{41} \oplus x_{42} \oplus x_{43}$.

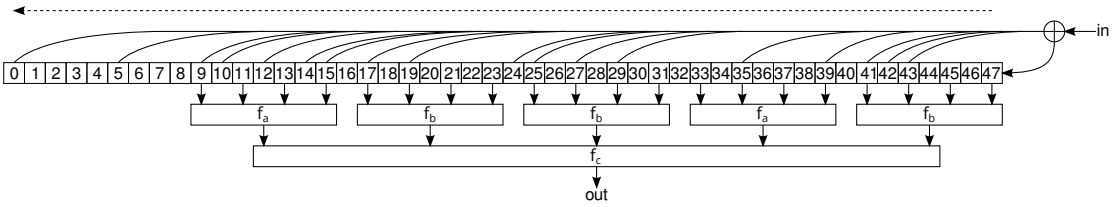


FIGURE 3.4.1: Structure of the CRYPTO1 stream cipher

Definition 3.4.2. The filter function $f\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by

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

Here $f_a, f_b\colon \mathbb{F}_2^4 \to \mathbb{F}_2$ and $f_c\colon \mathbb{F}_2^5 \to \mathbb{F}_2$ are defined by $f_a(y_0, y_1, y_2, y_3) := ((y_0 \vee y_1) \oplus (y_0 \wedge y_3)) \oplus (y_2 \wedge ((y_0 \oplus y_1) \vee y_3))$, $f_b(y_0, y_1, y_2, y_3) := ((y_0 \wedge y_1) \vee y_2) \oplus ((y_0 \oplus y_1) \wedge (y_2 \vee y_3))$, and $f_c(y_0, y_1, y_2, y_3, y_4) := (y_0 \vee ((y_1 \vee y_4) \wedge (y_3 \oplus y_4))) \oplus ((y_0 \oplus (y_1 \wedge y_3)) \wedge ((y_2 \oplus y_3) \vee (y_1 \wedge y_4)))$.

Because $f(x_0x_1\dots x_{47})$ only depends on $x_9, x_{11}, \dots, x_{47}$, we shall overload notation and see f as a function $\mathbb{F}_2^{20} \to \mathbb{F}_2$, writing $f(x_0x_1\dots x_{47})$ as $f(x_9, x_{11}, \dots, x_{47})$.

Definition 3.4.3. The pseudo-random generator feedback function $L_{16}\colon \mathbb{F}_2^{16} \to \mathbb{F}_2$ is defined by

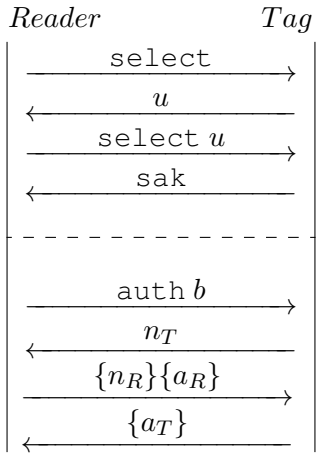
$$L_{16}(x_0x_1\dots x_{15}) := x_0 \oplus x_2 \oplus x_3 \oplus x_5.$$

Definition 3.4.4. The successor function $\mathrm{suc}\colon \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ is defined by

$$\mathrm{suc}(x_0x_1\dots x_{31}) := x_1x_2\dots x_{31}L_{16}(x_{16}x_{17}\dots x_{31})\,.$$

3.5 Authentication Protocol and Initialization

The authentication protocol was reverse engineered in [GKGM+08]. During the anti-collision phase, the tag is selected and sends its UID u to the reader. Then, the reader asks to authenticate for a specific memory block b. Consequently, the tag sends a challenge n_T. From this point on, the communication is encrypted, i.e. XOR-ed with the keystream. The reader responds with its own challenge n_R followed by the answer $a_R = \mathrm{suc}^{64}(n_T)$ to the tag challenge. The authentication is concluded with the tag answer $a_T = \mathrm{suc}^{96}(n_R)$. At this point, both the reader and tag are authenticated.



During the authentication protocol, the internal state of the stream cipher is initialized. Initially, the state is set to the sector key. Then, the challenge generated by the tag, n_T, is XORed with the uid u, and fed into the internal state and the feedback is applied accordingly. Subsequently, n_R, the nonce generated by the reader, is fed and feedback is applied. Since the communication is encrypted starting from n_R, the latter bits of n_R are influenced by the former bits of n_R. See Definition 3.5.1 for a more formal description of the initialization process.

We define the LFSR-*stream*, which allows us to conveniently address internal states as a whole and individual bits at any point in time. Below we define the LFSR-stream

$a_0a_1\ldots$ and keystream $ks_0ks_1\ldots$. We broadly employ the same notation as in [GRVS09], which is where the definitions are taken from. We will be using these definitions extensively throughout this thesis.

Definition 3.5.1. Given a key $k \in \mathbb{F}_2^{48}$, a tag nonce $n_T \in \mathbb{F}_2^{32}$, a UID $u \in \mathbb{F}_2^{32}$, and a reader nonce $n_R \in \mathbb{F}_2^{32}$, the internal state of the cipher at time i is $\alpha_i := a_i a_{i+1} \ldots a_{i+47} \in \mathbb{F}_2^{48}$.

Here the $a_i \in \mathbb{F}_2$ are given by

$$\begin{aligned} a_i &:= k_i && \forall i \in [0, 47] \\ a_{48+i} &:= L(a_i, \ldots, a_{47+i}) \oplus n_{T_i} \oplus u_i && \forall i \in [0, 31] \\ a_{80+i} &:= L(a_{32+i}, \ldots, a_{79+i}) \oplus n_{R_i} && \forall i \in [0, 31] \\ a_{112+i} &:= L(a_{64+i}, \ldots, a_{111+i}) && \forall i \in \mathbb{N}. \end{aligned}$$

Furthermore, we define the keystream bit $ks_i \in \mathbb{F}_2$ at time i by

$$ks_i := f(a_i a_{1+i} \ldots a_{47+i}) \qquad \forall i \in \mathbb{N}.$$

We denote encryptions by $\{\cdot\}$ and define $\{n_{R_i}\}, \{a_{R_i}\} \in \mathbb{F}_2$ by

$$\begin{aligned} \{n_{R_i}\} &:= n_{R_i} \oplus ks_{32+i} && \forall i \in [0, 31] \\ \{a_{R_i}\} &:= a_{R_i} \oplus ks_{64+i} && \forall i \in [0, 31]. \end{aligned}$$

Note that the $a_i, \alpha_i, ks_i\{n_{R_i}\}$, and $\{a_{R_i}\}$ are formally functions of k, n_T, u, and n_R. Rather than making this explicit by writing, e.g. $a_i(k, n_T, u, n_R)$, we just write a_i, where k, n_T, u, and n_R are clear from the context.

3.5.1 Nested authentication

When a reader is authenticated for a sector and hence communicating encrypted, a subsequent authentication request for another sector is also sent encrypted. At this point, the internal state of the cipher is initialized with the new key, which corresponds with the sector where the authentication is requested for. Furthermore, the authentication protocol is slightly different, since tag challenge n_T is also sent encrypted, i.e. $\{n_T\}$. Concretely, the initialization is similar to Definition 3.5.1, except that the bits of $\{n_T\}$ are decrypted before they are loaded into the internal state as shown in Definition 3.5.2. We refer to this procedure as a *nested authentication*. The attack proposed in this thesis only concerns the nested authentication.

Definition 3.5.2. In the situation from Definition 3.5.1, we define $\{n_{T_i}\} \in \mathbb{F}_2$ by $\{n_{T_i}\} := n_{T_i} \oplus ks_i \forall i \in [0, 31]$

Chapter 4

Known Vulnerabilities

In this chapter we highlight the various MIFARE Classic vulnerabilities which are described in the literature.

4.1 Short Key Length

The key size of 48 bits is too small to prevent a successful brute force attack within reasonable time. Initially, this was compensated for by the delay introduced by the communication and authentication procedure. Every attempt would take about 6 milliseconds. Hence, an online attack on a single card for a single sector would take more than 44 thousand years, searching through all 2^{48} possible keys. However, when the CRYPTO1 algorithm was exposed, an offline brute force attack could be mounted, eliminating the delay caused by the communication with the card. In 1996 it was already strongly recommended against using symmetric cryptosystems that use 56 bits keys [BDR+96]. Nohl and Plötz stated in December 2007 [NP07] that a $100 key cracker will find a key in approximately 1 week. They claim that it can even be done much faster when trading memory for time.

4.2 Predictable Nonces

It is commonly known that proper pseudo-random number generators are essential for cryptographic protocols to provide proper security. The 32 bits nonce used by the MIFARE Classic is generated by a 16 bit LFSR, meaning the entropy of the nonces is only 16 bits, which is clearly insufficient. Given its structure, the sequence of nonces is repeated every $2^{16} - 1$ cycles. On top of that, whenever the card is powered up, the LFSR is reset to a known state. Hence, if an attacker carefully keeps the time constant between powering up the card and requesting a nonce, the nonce will be constant. This essentially removes all the randomness introduced by the card from the authentication.

This weakness is exploited in many of the known attacks. Furthermore, besides the attack described in Chiu et al. [CHC+14], all card-only attacks proposed in the literature exploit this particular implementation mistake.

The attack presented in this thesis does not depend on this vulnerability since our goal is to devise an attack which works on all MIFARE Classic compatible cards, including those that use a proper PRNG.

4.3 The Nested Authentication

Once a single key of a single sector is known, an attacker can authenticate against that sector, and while communicating (encrypted) with the tag, send another authentication

request for a different sector and/or key. When this authentication command has been processed, the cipher's internal state is set to the key of the new sector and the authentication protocol depicted in Section 3.5 starts again. This time, though, the challenge generated by the tag is also sent encrypted, using the key for the sector that the authentication is requested for.

In case the card has the weak pseudo-random number generator vulnerability mentioned above, and hence allowing an attacker to predict the nonce, the nested authentication can be used to recover 32 bits of keystream by only wirelessly interacting with a card. This phenomenon was exploited in [GRVS09] in one of their attacks, which is referred to as the *nested attack*.

The attack presented in this thesis relies on the nested authentication, since, assuming are concerned with a hardened card and with no access to a genuine reader, it is the only channel through which secret key information is leaked.

4.4 Parity bits

The MIFARE Classic sends a parity bit for each byte it transmits. Contrary to the ISO/IEC 14443-A standard [ISO01], the data link layer and communication layer are mixed. Rather than computing parity bits over the bits that are sent over the air, i.e., the ciphertext, they are computed over the plaintext. On top of that, the internal state does not shift during the encryption of parity bits, i.e. the keystream bit used to encrypt a parity bit is re-used to encrypt the next bit of plaintext. See Figure 4.4.1 for an illustration of this property.

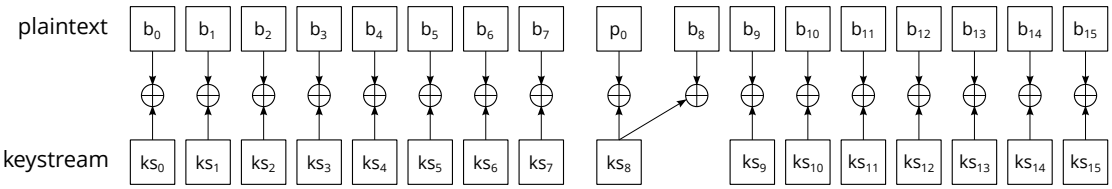


FIGURE 4.4.1: The encryption of parity bits

Given this property an attacker can learn information about the plaintext by observing only the ciphertext. Consequently, this already breaks the confidentiality of the encryption scheme. For example: suppose we observed the encrypted parity bit $\{p_0\}$, which is computed over the first nonce byte $n_{T_{[0,7]}}$, and $\{n_{T_8}\}$, the first bit of the second encrypted nonce byte. Since both are encrypted (XOR-ed) with the same keystream bit ks_8, we can deduce whether or not the plaintext parity p_0 equals n_{T_8}.

In this research, we focus only on the parity bits of $\{n_T\}$. The ISO standard specifies odd parity, hence the "$\oplus 1$" in the definition below.

Definition 4.4.1. In the situation from Definition 3.5.1, we define the parity bits $p_j \in \mathbb{F}_2$ by

$$p_j := \bigoplus_{i=0}^{7} n_{T_{8j+i}} \oplus 1 \qquad \forall j \in [0,3]$$

and the encryptions $\{p_j\}$ of these parity bits by

$$\{p_j\} := p_j \oplus ks_{8j+8} \qquad \forall j \in [0,3]$$

The only way left in hardened cards that allows an attacker to observe keystream information, without the need for communicating with a genuine reader, is through parity bits. Hence, the attack presented in this thesis is built upon this vulnerability.

4.5 Keystream leakage through errors

While the authentication protocol is running, the card always first checks the parity bits before doing anything else. Hence, during the authentication protocol, when the card receives $\{n_R\}$ and $\{a_R\}$, if at least one of the eight parity bits is wrong, the card does not respond. In the case all eight parity bits are correct, but the answer a_R is wrong, the card replies with the 4-bit error code `0x5`, indicating a failed authentication. The error code is sent encrypted, even though the reader has not successfully authenticated itself and hence cannot be assumed to be able to decrypt it correctly.

From this encrypted error code, 4 bits of keystream is leaked and can be obtained by XOR-ing the encrypted error code with its plaintext value. The leakage of 4 bits of keystream may not seem a severe issue. However, the leakage is a crucial ingredient for several card-only key recovery attacks [GRVS09, Cou09, CHC+14].

The weakness can be mitigated by issuing cards that do not send authentication error codes. This does not break compatibility with the MIFARE Classic protocol since, as stated, the reader is unable to decrypt it anyway.

4.6 LFSR Rollback

Initially, the cipher's internal state is set to the secret key. During authentication and during encryption, the state is updated. However, given an internal state at any given point in time, and given the data fed into the LFSR, u, n_T and $\{n_R\}$, the previous state can be computed deterministically. Hence, it is sufficient for an attacker to obtain the internal state at any point in time. The LFSR state can then be rolled back to time 0, wherein it holds the secret key. Many, if not all attacks rely on this weakness, including the one presented in this thesis. The fact that this vulnerability is present in MIFARE Classic was first pointed out by Garcia et al. [GKGM+08].

4.7 Odd Inputs to the Filter Function

The nonlinear filter function f that takes inputs from the LFSR to produce a keystream bit exclusively uses bits on odd-numbered positions, i.e. $a_9, a_{11}, a_{13}, \ldots, a_{47}$ (see Figure 3.4.1). The fact that they are so evenly placed can be exploited. Given a part of keystream, the bits of the LFSR relevant for generating the even and odd bits of that keystream can be generated separately. By splitting the feedback into two parts as well, those even and odd states can be combined efficiently to recover exactly those LFSR states that produce a given keystream. This reduces the computing power required for an exhaustive search from 2^{39} to approximately $2^{20} + 2^{19} \approx 2^{20.58}$. It may be understood as "inverting" the filter function f [GKGM+08]. The attack presented in this thesis used this vulnerability extensively. The complete attack is described in full detail in the next chapter.

Chapter 5

Attacking Mifare Classic

In this chapter we describe the process of mounting an attack without exploiting the two main implementation mistakes: the weak pseudo random number generator (Section 4.2) and the encrypted error message (Section 4.5). We propose a novel attack that solely depends on design issues in the CRYPTO1 cipher. Hence, in order to avoid this attack, backwards compatibility with the MIFARE Classic protocol must be broken.

Stage 1 We start retrieving encrypted nonces $\{n_T\}$ using the nested authentication, i.e. by authenticating for a sector for which the key is already known, followed by an authentication request for the target sector. This process is repeated in the background until the key of the target sector is recovered.

Stage 2 Given the set of encrypted nonces we have obtained so far, we determine $\mathcal{S}_\epsilon$, the *sum property* of the cipher's initial state. This property is explained in detail in Section 5.2 We also determine $\mathcal{S}_{\{b\}}$, the sum property of the cipher's state after byte $\{b\}$ is fed (i.e. at time 8), for all 256 possible first input bytes $\{b\}$. Depending on the probability that we guessed $\mathcal{S}_{\{b\}}$ correctly, we choose to incorporate byte $\{b\}$ in the differential analysis described in Section 5.5. This is done using a probability threshold value. How the probability of correctly guessing the sum property for input byte $\{b\}$ is computed is explained in detail in Section 5.4. Additionally, we incorporate all input bytes $\{b\}$ for which we observe that $f(\alpha_8) \neq f(\alpha_8 \oplus 1)$, i.e. all first nonce bytes for which the filter flip property holds. This is explained in Section 5.6.

Next, given the information determined from the set of encrypted nonces, we determine the size of the leftover search space. The leftover search space shrinks as the number of harvested encrypted nonces increases since more nonces allow us to more accurately guess sum properties and observe filter flip properties. We repeat this information gathering step until the search space is sufficiently small, as subjectively assessed by the attacker. Once this is the case, we move on to Stage 3.

Stage 3 Given the information determined from the set of encrypted nonces so far, we construct a candidate list for $a_{[9,55]}$ (Section 5.5). Which we extend to $a_{[8,55]}$ by prefixing both 0 and 1. Then, we perform an LFSR-rollback described in Section 4.6 to transform them into candidates for $a_{[0,47]}$, i.e. the secret key. Subsequently, we carry out the offline brute force attack presented in Section 5.1. The key is not always found since sum properties are guessed correctly with high probabilities, not certainty. In case the key is not found, we revert to Stage 2, optionally with an increase of the probability threshold, causing the search space to increase. However, gathering of more nonces increases the certainty and reduces the number of candidate keys.

5.1 Offline Brute Force Attack

Recall from Definition 4.4.1 that a parity bit p_i is computed over plaintext byte $n_{T_{[8i,8i+7]}}$, and subsequently encrypted (XOR-ed) with the next keystream bit ks_{8i+8}. This property can be exploited in order to verify whether a candidate key is the correct key. Given an encrypted nonce obtained through a nested authentication attempt, the attacker can attempt to "decrypt" the nonce using the candidate key. In case the candidate is the correct key, the parity bits will be correct. However, in case a wrong key was used, a parity bit will be correct with probability $\frac{1}{2}$.

An encrypted nonce holds 4 bytes, thus 4 encrypted parity bits. Therefore, on average, $48/4 = 12$ encrypted nonces are enough to determine the key uniquely.

We implemented this brute-force attack and executed it on an NVIDIA GTX460 GPU. From our experiments we deduced that performing a full 48-bit exhaustive search takes approximately 1 month. However, our implementation leaves headroom for optimizations, such as bitslicing [Bih97]. We assume that such optimizations improve the attack performance by at least a factor of four.

By today's standards the GTX460 is considered a low end GPU. It has 336 cores and runs at 675 Mhz. Clearly, the attack scales linearly in the amount of parallel computing power available. As of today, a GTX460 costs approximately $50 USD. Therefore, its price/performance ratio is among the best available. We reserve another $20 USD per GPU for hardware driving the GPUs (CPUs, mainboards, power supplies). Hence, recovering a single key within an hour by means of a brute force attack would require a hardware budget of approximately $12,600 USD.

The attack mounted in [NP07] seems significantly faster. However, this is mainly because it operates directly on the keystream, while the brute force attack described here operates on contiguously reinitializing the state with encrypted nonces.

Although we have not researched the possibility to mount a time-memory trade-off extensively, on the surface, doing so seems very difficult and comes with negligible performance impact. This is due to the fact that time-memory trade-off strategies found in scientific literature [Hel80, BS00] operate on keystream, while our attacker model dictates that we can only deduce indirect properties of the keystream. On top of that, the cipher's internal state is initialized by a random nonce which can not be influenced by an attacker.

In the next sections we describe several properties that can be observed by only analyzing the ciphertext. Once the properties are determined, all candidate keys for which these properties do not hold can be discarded, significantly reducing the search space.

Note that an exhaustive search and subsequently testing each candidate for the properties found is a computationally expensive task, since an offline brute force attack is rather slow. Therefore, we also introduce an efficient method for constructing a list of key candidates in Section 5.3.

5.2 Sum Property

The first property that can be observed by analyzing the ciphertext is the sum property. In order to define it, we first need to establish a number of lemmas.

Given an encrypted nonce byte $\{n_{T_{[8i,8i+7]}}\}$ and corresponding encrypted parity bit $\{p_i\}$. We can cancel out the plaintext by XOR-ing the two together, yielding the XOR of the individual keystream bits $ks_{8i} \ldots ks_{8i+8}$.

Lemma 5.2.1. *Let $\{n_{T_{[8i,8i+7]}}\}$ be the i^{th} encrypted nonce byte obtained from the card and let $\{p_i\}$ be its corresponding encrypted parity bit. Then*

$$\bigoplus_{j=0}^{7}\{n_{T_{8i+j}}\} \oplus \{p_i\} = \bigoplus_{j=0}^{8} ks_{8i+j} \oplus 1$$

Proof.

Take the XOR of the individual bits of the nonce byte and XOR it together with parity bit

$$\bigoplus_{j=0}^{7}\{n_{T_{8i+j}}\} \oplus \{p_i\}$$

Definition of parity bit

$$= \bigoplus_{j=0}^{7}\{n_{T_{8i+j}}\} \oplus \{\bigoplus_{j=0}^{7} n_{T_{8i+j}} \oplus 1\}$$

Encrypted byte is plaintext XOR-ed with keystream

$$= \bigoplus_{j=0}^{7}(n_{T_{8i+j}} \oplus ks_{8i+j}) \oplus \bigoplus_{j=0}^{7} n_{T_{8i+j}} \oplus 1 \oplus ks_{8i+8}$$

Split leftmost XOR notation

$$= \bigoplus_{j=0}^{7} n_{T_{8i+j}} \oplus \bigoplus_{j=0}^{7} ks_{8i+j} \oplus \bigoplus_{j=0}^{7} n_{T_{8i+j}} \oplus 1 \oplus ks_{8i+8}$$

$\bigoplus_{j=0}^{7} n_{T_{8i+j}}$ *is XORed twice, hence removed*

$$= \bigoplus_{j=0}^{8} ks_{8i+j} \oplus 1$$

□

The lemma shown above is applicable to any ciphertext, e.g. it can also be applied to $\{n_R\}$ or $\{a_R\}$. However, in this thesis we are only concerned with the analysis of $\{n_T\}$. The following lemma states that for two encrypted nonces with a common prefix of j bits, the LFSR-stream $a_{48}a_{49}\ldots a_{48+j}$ is equal.

Lemma 5.2.2. *Suppose that we have two encrypted nonces $\{n_T\} := \{n_{T_0}n_{T_1}\ldots n_{T_{31}}\}$ and $\{n'_T\} := \{n'_{T_0}n'_{T_1}\ldots n'_{T_{31}}\}$, and their corresponding LFSR-streams $a_0a_1\ldots$ and $a'_0a'_1\ldots$ for the same key. Let the encrypted nonces have a common prefix of j bits, i.e. $\{n_{T_i}\} = \{n'_{T_i}\}$, for all $i < j$. Then also $a_{48+i} = a'_{48+i}$ and $n_{T_i} = n'_{T_i}$ for $i < j \leq 31$. Furthermore, if $\{n_{T_j}\} \neq \{n'_{T_j}\}$, then $a_{48+j} \neq a'_{48+j}$ and $n_{T_j} \neq n'_{T_j}$.*

Proof. By induction

Base case : $a_{[0,47]}$

Trivially, since $a_{[0,47]}$ is the key, it is equal to $a'_{[0,47]}$.

Step : $a_{[48,79]}$

Following Definitions 3.5.1 and 3.5.2, we get for $0 \leq i < 32$:

$$\begin{aligned} ks_i &= f(a_{9+i}a_{11+i}\dots a_{47+i}), \\ n_{T_i} &= \{n_{T_i}\} \oplus ks_i \\ z_i &= ks_i \oplus L(a_{0+i}a_{5+i}\dots a_{43+i}) \oplus u_i \\ a_{48+i} &= \{n_{T_i}\} \oplus z_i \end{aligned}$$

Given that $a'_{[0,47+i]} = a'_{[0,47+i]}$ and the fact that the uid u is constant, we obtain $ks_i = ks'_i$ and $z_i = z'_i$. Since $\{n_{T_i}\} = \{n'_{T_i}\}$, we obtain $n_{T_i} = n'_{T_i}$ and $a_{48+i} = a'_{48+i}$.

Suppose that $\{n_{T_j}\} \neq \{n'_{T_j}\}$. We re-use the inductive step above with $i = j$. Given that $a_{[0,47+j]} = a'_{[0,47+j]}$ and the fact that uid u is constant, we obtain $ks_j = ks'_j$ and $z_j = z'_j$. Since $\{n_{T_j}\} \neq \{n'_{T_j}\}$, we obtain $n_{T_j} \neq n'_{T_j}$ and $a_{48+j} \neq a'_{48+j}$. □

Finally, we describe another property that, when taken together with Lemma 5.2.1, allows us to define the desired sum property. Let's focus on any of the encrypted nonce bytes $\{n_{T_{[8i,8i+7]}}\}$ where $i \in [0,3]$. The value of this byte is mapped to the LFSR-stream byte $a_{[8i+48,8i+55]}$. The value is not taken directly, but first mangled by the keystream and feedback loop. When $\{n_{T_{[0,8i-1]}}\}$ is constant, a one-to-one mapping between $\{n_{T_{[8i,8i+7]}}\}$ and $a_{[8i+48,8i+55]}$ exists.

Lemma 5.2.3. *Given that $\{n_{T_{[0,8h-1]}}\}$ is constant, a one-to-one mapping exists between all 256 possible values for nonce byte $\{n_{T_{[8h,8h+7]}}\}$ and LFSR-stream byte $a_{[8h+48,8h+55]}$, where $h \in [0,3]$.*

Proof. Suppose we have two encrypted nonces $\{n_T\} := \{n_{T_0}n_{T_1}\dots n_{T_{31}}\}$ and $\{n'_T\} := \{n'_{T_0}n'_{T_1}\dots n'_{T_{31}}\}$, and their corresponding LFSR-streams $a_0a_1\dots$ and $a'_0a'_1\dots$. The first $h-1$ bytes of both nonces are equal, i.e. $\{n_{T_{[0,8h-1]}}\} = \{n'_{T_{[0,8h-1]}}\}$, and the h^{th} nonce byte differs, i.e. $\{n_{T_{[8h,8h+7]}}\} \neq \{n'_{T_{[8h,8h+7]}}\}$. Let $\{n_{T_i}\}$ be the i^{th} bit of $\{n_T\}$, such that $\{n_{T_i}\} \neq \{n'_{T_i}\}$, and $\{n_{T_j}\} = \{n'_{T_j}\}$, for all $j < i$. By definition, this bit is contained within $\{n_{T_{[8h,8h+7]}}\}$.

By Lemma 5.2.2, we obtain $a_{48+j} = a'_{48+j}$ for all $j < i$, and $a_{48+i} \neq a'_{48+i}$. Hence, $\{n_{T_{[8h,8h+7]}}\}$ and $\{n'_{T_{[8h,8h+7]}}\}$ are never mapped to the same $a_{[8h+48,8h+55]}$. Furthermore, since there are 256 possible values for both $\{n_{T_{[8h,8h+7]}}\}$ and $a_{[8h+48,8h+55]}$, the mapping must be one-to-one.

□

Suppose we have collected sufficiently many encrypted nonces such that we observed all 256 possible values for an encrypted nonce byte $\{n_{T_{[8i,8i+7]}}\}$, given all previous bytes $\{n_{T_{[0,8i-1]}}\}$ are constant. This information allows us to compute the *sum property*.

Definition 5.2.4. The *sum property*, $\mathcal{S}$ is a property of the internal state at time $8i$, where $i \in [0,3]$, that can be observed by retrieving all possible values for $\{n_{T_{[8i,8i+7]}}\}$, i.e. the i^{th} encrypted nonce byte, with $\{n_{T_{[0,8i-1]}}\}$, i.e. all previous encrypted nonce bytes, being constant. For each encrypted nonce byte we take the XOR over its individual bits, XOR it together with its corresponding parity bit and negate the result. Next, we take the

sum over all the resulting values, ignoring the modulo operation of finite fields. The result is a number ranging from 0 until and including 256.

$$\mathcal{S}(\alpha_{8i}) := \sum_{j=0}^{255}(\bigoplus_{h=0}^{7}\{n_{T_{8i+h}}\} \oplus \{p_i\} \oplus 1), \qquad \text{where } \{n_{T_{[8i,8i+7]}}\} = j$$

The sum property is equivalent to a property that depends only on the cipher's internal state at time $8i$. The following lemma formalizes this.

Lemma 5.2.5. *Suppose that $\{n_{T_{[0,8i-1]}}\}$ is constant. Then, sum property $\mathcal{S}$ is equivalent to the following function over the cipher's internal state*

$$\mathcal{S}(\alpha_{8i}) = \sum_{j=0}^{255}(\bigoplus_{h=0}^{8} f(a_{8i+h+9}a_{8i+h+11}\dots a_{8i+h+47})), \qquad \textit{where } a_{[8i+48,8i+55]} = j$$

Proof.

Take the XOR of the individual bits of encrypted nonce byte $\{n_{T_{[8i,8i+7]}}\}$ and XOR it together with its corresponding encrypted parity bit. Take the sum over the result for all possible encrypted nonce bytes.

$$\sum_{j=0}^{255}(\bigoplus_{h=0}^{7}\{n_{T_{8i+h}}\} \oplus \{p_i\} \oplus 1), \qquad \text{where } \{n_{T_{[8i,8i+7]}}\} = j$$

Apply lemma 5.2.1

$$= \sum_{j=0}^{255}(\bigoplus_{h=0}^{8} ks_{8i+h}), \qquad \text{where } \{n_{T_{[8i,8i+7]}}\} = j$$

Previous nonce bytes are constant (given), hence Lemma 5.2.3 applies, and the order is irrelevant for $\sum$

$$= \sum_{j=0}^{255}(\bigoplus_{h=0}^{8} ks_{8i+h}), \qquad \text{where } a_{[8i+48,8i+55]} = j$$

Definition of ks_i

$$= \sum_{j=0}^{255}(\bigoplus_{h=0}^{8} f(a_{8i+h+9}a_{8i+h+11}\dots a_{8i+h+47})), \qquad \text{where } a_{[8i+48,8i+55]} = j$$

□

Because $\mathcal{S}(a_{[8i,8i+47]})$ only depends on $a_{[8i+9,8i+47]}$, we shall overload notation and see $\mathcal{S}$ as a function $\mathbb{F}_2^{39} \to [0, 256]$, writing $\mathcal{S}(a_{[8i,8i+47]})$ as $\mathcal{S}(a_{[8i+9,8i+47]})$.

Note that, at this point, we can already reduce the exhaustive search space significantly by collecting all possible values for the first encrypted nonce byte, which determine the sum property of the cipher's initial state, i.e. at time 0, and discard all states having a sum property value different from the one observed by means of precomputed lookup tables. In the next section we show how to do this efficiently and without lookup tables.

5.3 Splitting the Sum property

In Section 4.7 we described a vulnerability of the filter function f, namely that all its inputs are evenly placed at odd positions. In this section we show how we can exploit

this property to construct candidate key lists such that each candidate has a certain given value for the sum property, without the need for going through the entire 39-bit search space and testing for this property.

To do so, we first introduce the *partial sum property*, which is the sum taken over only the odd or even bits of the cipher's internal state.

Definition 5.3.1. The *odd sum*, $\mathcal{S}_O$ depicts the sum property computed over the odd bits of the cipher's internal state. Likewise, $\mathcal{S}_E$ depicts the *even sum*. Both values range from 0 until and including 16. Note that both values are do not share a single input, hence they are completely independent of one another. They are defined as follows.

$$\mathcal{S}_O(a_9 a_{11} \dots a_{47}) := \sum_{j=0}^{15} \bigoplus_{i=0}^{4} f(a_{9+2i} a_{11+2i} \dots a_{47+2i}), \qquad \text{for } (a_{49} a_{51} a_{53} a_{55}) = j$$

$$\mathcal{S}_E(a_{10} a_{12} \dots a_{46}) := \sum_{j=0}^{15} \bigoplus_{i=0}^{3} f(a_{10+2i} a_{12+2i} \dots a_{48+2i}), \qquad \text{for } (a_{48} a_{50} a_{52} a_{54}) = j$$

Note that, in order to improve readability, we have implicitly assumed we are concerned with the first encrypted nonce byte. However, this need not be the case. In fact, partial sum properties can be computed over any nonce byte, as long as all previous bytes are kept constant, i.e. the same prerequisites hold as with the ordinary sum property.

Next, we define sets consisting of all possible internal state bits T_{O_p}, T_{E_q} determined by partial sum property p, q for the odd and even sum property, respectively.

$$T_{O_p} := \{x \mid x \in \mathbb{F}_2^{20} \text{ and } \mathcal{S}_O(x) = p\}$$
$$T_{E_q} := \{x \mid x \in \mathbb{F}_2^{19} \text{ and } \mathcal{S}_E(x) = q\}$$

Now suppose that we have determined both the odd and even sum property of the cipher's internal state. As we will see below, this uniquely determines the sum property.

Lemma 5.3.2. *Suppose that both $\mathcal{S}_O$ and $\mathcal{S}_E$ of the cipher's internal state are known. Then also $\mathcal{S}$ is known and is given by the following equation.*

$$s = p(16 - q) + (16 - p)q, \text{ where } p, q, s = \mathcal{S}_O(\cdot), \mathcal{S}_E(\cdot), \mathcal{S}(\cdot)$$

Proof. We take $\mathcal{S}_O(\cdot), \mathcal{S}_E(\cdot), \mathcal{S}(\cdot) = p, q, s$, respectively.

Firstly, we take the definition of s itself. For brevity we write ks_i for $f(a_{9+i} a_{11+i} \dots a_{47+i})$.

$$s = \sum_{j=0}^{255} \bigoplus_{i=0}^{8} ks_i, \text{ where } a_{[48,55]} = j$$

Next, we remove the $\sum$ notation and write the equation in full

$$\begin{aligned} s \; = & \bigoplus_{i=0}^{8} ks_i, \text{ [where } a_{[48,55]} = 0] + \\ & \bigoplus_{i=0}^{8} ks_i, \text{ [where } a_{[48,55]} = 1] + \\ & \vdots \\ & \bigoplus_{i=0}^{8} ks_i, \text{ [where } a_{[48,55]} = 255] \end{aligned}$$

We split up the $\bigoplus$ notation and rewrite the equation as follows

$$
\begin{aligned}
s = \\
&\left(\begin{array}{l}
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 0]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 0]) + \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 0]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 1]) + \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 0]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = 15])
\end{array}\right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \\
&\left(\begin{array}{l}
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 1]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 0]) + \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 1]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 1]) + \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = \ 1]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = 15])
\end{array}\right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + \\
&\left(\begin{array}{l}
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = 15]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 0]) + \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = 15]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = \ 1]) + \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
(\bigoplus_{i=0}^{4} ks_{2i},\ [\text{where } a_{49}a_{51}a_{53}a_{55} = 15]) \oplus (\bigoplus_{i=0}^{3} ks_{1+2i},\ [\text{where } a_{48}a_{50}a_{52}a_{54} = 15])
\end{array}\right)
\end{aligned}
$$

The definitions of p and q are the following

$$
\begin{aligned}
p &= \sum_{j=0}^{15} \bigoplus_{i=0}^{4} ks_{2i}, && \text{where } a_{49}a_{51}a_{53}a_{55} = j \\
q &= \sum_{j=0}^{15} \bigoplus_{i=0}^{3} ks_{1+2i}, && \text{where } a_{48}a_{50}a_{52}a_{54} = j
\end{aligned}
$$

If we examine the last representation of s and focus only on a single part between large brackets, choosing any value for $a_{49}a_{51}a_{53}a_{55}$, then we see that the value for $\bigoplus_{i=0}^{4} ks_{2i}$ is negated exactly q times and $16 - q$ times taken directly. Generalizing this for all 16 possible values for $a_{49}a_{51}a_{53}a_{55}$ yields

$$s \ = \ p(16 - q) + (16 - p)q$$

□

Note that the proof above is only concerned with the sum property of the first encrypted nonce byte. However, the lemma also applies to subsequent nonce bytes, given that previous nonce bytes are constant. However, generalizing the proof in order to take this into account highly impacts simplicity and readability. Therefore we chose to avoid it.

One may expect the sum property value to appear normally distributed by its definition. However, this is not the case due to, among other phenomena, the property just described. See Figure 5.3.1 for a bar chart depicting the possible values for the sum property against their corresponding probabilities for a randomly chosen cipher state.

At this point, we have all the building blocks needed for constructing a candidate key list without iterating through the entire 39-bit search space and subsequently testing for the sum property and without using (large) precomputation tables. To do so:

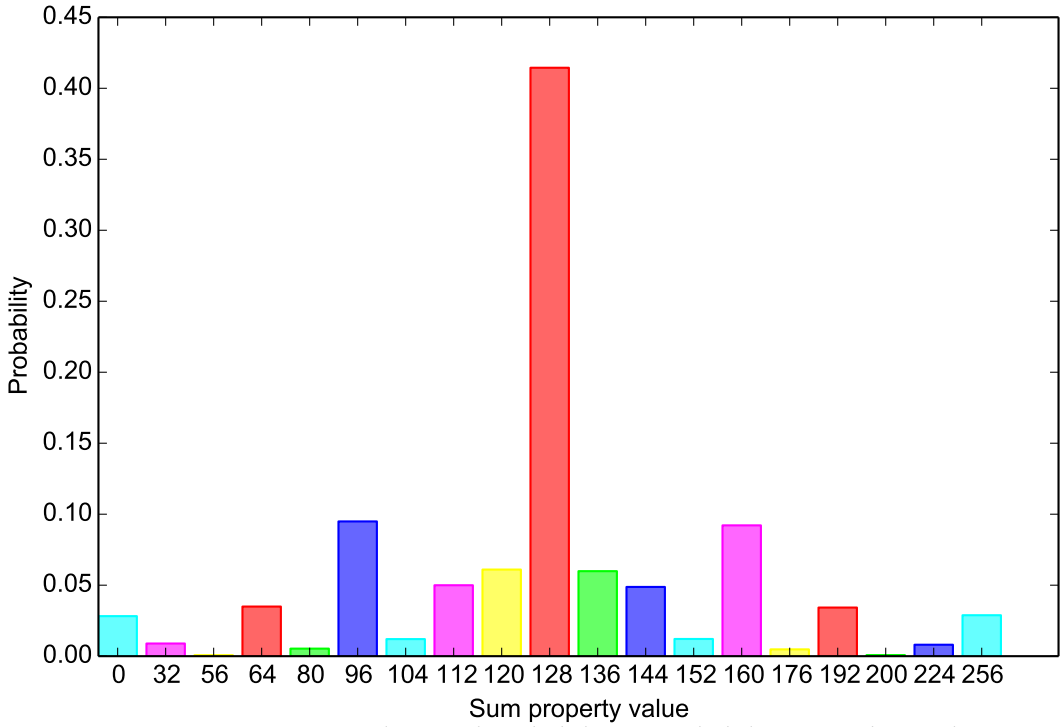


FIGURE 5.3.1: Sum values and probabilities. Probabilities are obtained by averaging over 8192 random cipher states

(i) We generate all tables T_{O_p} and T_{E_q}, where $p, q \in [0, 16]$, consisting of all possible odd and even LFSR bits used by their corresponding partial sum function. Precomputation is not necessary as computation completes within one second on an ordinary laptop.

(ii) We determine $s = \mathcal{S}(a_{[9,47]})$, the sum property of the initial LFSR state by retrieving all 256 possible values for the first encrypted nonce byte $n_{T_{[0,7]}}$, and subsequently apply Lemma 5.2.5.

(iii) Once s has been determined, we take all possible combinations for $(p, q) \in [0, 16] \times [0, 16]$ such that $s = p(16 - q) + (16 - p)q$ holds.

(iv) For each such a combination, we take all values $x \in T_{O_p}$ and combine them with all values $y \in T_{E_q}$. We define $z \in \mathbb{F}_2^{39}$, $z := x_0 y_0 x_1 y_1 \dots x_{18} y_{18} x_{19}$. By construction $\mathcal{S}(z) = \mathcal{S}(a_{[9,47]})$. Therefore, every z is a candidate for 39 bits of the cipher's internal state. Lemma 5.3.2 states the sum property is defined by its partial sum properties. Hence, a value z must exist such that $z = a_{[9,47]}$.

In the resulting candidate list, each entry holds 39 bits of LFSR state, whereas the length of this list is given by the probability of the observed sum property times 2^{39}. On average the list holds $2^{36.72}$ entries, i.e. a drop of 2.28 bits in complexity. In the next section, we describe how we can efficiently determine the sum property at time 8.

5.4 Determining the sum property at time 8

As we have seen in the last section, by determining the sum property of the cipher's initial internal state, the exhaustive search space can be significantly reduced. We can apply the same technique once more, only this time we determine the sum property of

the cipher's internal state at time 8, i.e. after the first nonce byte is fed. As stated as a requirement in Lemma 5.2.5, in order to determine the sum property at time 8, we need to collect all possible values for the second nonce byte $n_{T_{[8,15]}}$, while the first byte $n_{T_{[0,7]}}$ is constant. We assume the encrypted nonce $\{n_T\}$ produced by the tag is random and beyond our control. Hence, we require

$$256 \cdot (\frac{256}{256} + \frac{256}{255} + \frac{256}{254} + \cdots + \frac{256}{1}) \approx 401365.07$$

nonces on average. This analysis relates to the well-known coupon collector problem [FGT92].

Obtaining this number of nonces would typically take time in the order of hours. However, by taking a probabilistic approach we can determine the sum property at time 8 with very high probability with much fewer nonces.

The relation between byte $\{n_{T_{[8,15]}}\}$ and $a_{[56,63]}$ is unknown to us since it depends on the key. However, according to Lemma 5.2.3, it is one-to-one, hence collision-free. Additionally, $\{n_T\}$, and thus $\{n_{T_{[8,15]}}\}$ is chosen randomly by the card. As such, we regard every unique $\{n_{T_{[8,15]}}\}$ we receive as a random (though unknown) sample for $a_{[56,63]}$.

Let U be a set of tuples $(\{b\}, \{p\}) \in \mathbb{F}_2^8 \times \mathbb{F}_2$ storing encrypted nonce bytes together with their corresponding encrypted parity bits. We define U such that it contains all unique samples we received for $\{n_{T_{[8,15]}}\}$, i.e. the second nonce byte, with a constant $\{n_{T_{[0,7]}}\}$.

We give a brief example in order to clarify the intuition behind the probabilistic approach: Suppose U has 20 entries. From these samples, we compute $k = \sum_{(\{b\},\{p\})\in U} \bigoplus_{i=0}^{7} \{b\} \oplus \{p\}$, i.e. the sum property over the sample space. Now suppose that we find $k = 0$. From this observation we may conclude that the sum property at time 8 with the prefix chosen is very likely 0. In fact, the actual probability for this to be the case is approximately 0.9775. Although we only have 20 samples for $\{n_{T_{[8,15]}}\}$. Hence, in this example, we require to gather only

$$256 \cdot (\frac{256}{256} + \frac{256}{255} + \frac{256}{254} + \cdots + \frac{256}{237}) \approx 5320.24$$

nonces on average. We now formalize this this intuition. A key concept we use here is the Hypergeometric distribution.

Definition 5.4.1. The hypergeometric distribution is a discrete probability distribution that describes the probability of k successes in n draws, *without* replacement, from a finite population of size N containing exactly K successes, wherein each draw is either a success or failure.

A random variable X_K follows the hypergeometric distribution if its probability mass function (pmf) is given by

$$P(X_K = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

Where

N	is the population size
K	is the number of success states in the population
n	is the number of draws
k	is the number of successes

We define the following random variables

$S :=$ The sum property
$T :=$ The result of $\sum_{(\{b\},\{p\})\in U} \bigoplus_{i=0}^{7}\{b\} \oplus \{p\}$, i.e. the sum property over the current sample space
$X_K :=$ The result of $\sum_{(\{b\},\{p\})\in U} \bigoplus_{i=0}^{7}\{b\} \oplus \{p\}$, *where the actual sum property* $\mathcal{S}(\alpha_8)$ *is equal to* K

X_K follows a hypergeometric distribution with parameters N, K, n and output k, where
$N =$ 256
$K =$ $\mathcal{S}(\alpha_8)$, i.e. the sum property at time 8
$n =$ $\#U$, i.e. the number of unique samples gathered for $\{n_{T_{[8,15]}}\}$, for some constant $\{n_{T_{[0,7]}}\}$
$k =$ $\sum_{(\{b\},\{p\})\in U} \bigoplus_{i=0}^{7}\{b\} \oplus \{p\}$, i.e. the sum property computed over the current sample space

By definition of the random variables we get

$$P(X_K = k) = P(T = k|S = K)$$

Our goal is to compute the probability of the sum property having a certain value, *given the sum property computed over the sample obtained so far*, hence $P(S = K|T = k)$. We use Bayes' theorem to obtain it.

$$P(S = K|T = k) = \frac{P(T = k|S = K)\, P(S = K)}{P(T = k)}$$

To obtain $P(T = k)$*, we take* $P(T = k|S = i)P(S = i)$ *for all possibilities for* i*. The events are mutually exclusive.*

$$P(S = K|T = k) = \frac{P(T = k|S = K)\, P(S = K)}{\sum_{i=0}^{256} P(T = k|S = i)P(S = i)}$$

Definition of X_K

$$P(S = K|T = k) = \frac{P(X_K = k)\, P(S = K)}{\sum_{i=0}^{256} P(X_i = k)P(S = i)}$$

Our strategy becomes to take every possible sum property value for K and compute $P(S = K|T = k)$. Our guess for $\mathcal{S}(\alpha_8)$ will be the K that yields the highest probability.

From the result we see that, in order to compute the probability for a sum property value given a sample, we need to know $P(S = i)$ for every possible value of i from 0 until and including 256. These probabilities are depicted in Figure 5.3.1.

5.5 Differential Analysis

From Section 5.4 we have seen that we can determine $\mathcal{S}(\alpha_0)$ and guess $\mathcal{S}(\alpha_8)$ for a given $\{n_{T_{[0,7]}}\}$ by gathering only a modest number of encrypted nonces. Additionally, once the value for $\mathcal{S}(\alpha_8)$ is known, Section 5.3 has shown that we can construct candidate lists containing 39 bits of LFSR state, without the need of going through all 2^{39} possible states, computing the sum property, and testing whether this results in the correct value. In this section, we show how we can combine two or more sum properties and eliminate a significant amount of impossible key candidates prior to constructing the final candidate list.

Given that we guessed $\mathcal{S}(\alpha_8)$ for a given $\{n_{T_{[0,7]}}\}$ with near certain probability, it is very likely that we can also guess this for other input bytes without requiring to collect additional nonces, since we assume the encrypted nonces generated by the card are

random and beyond our control. Input byte $\{n_{T_{[0,7]}}\}$ is mapped to $a_{[48,55]}$ and therefore affects $\mathcal{S}(\alpha_8)$. Hence, $\mathcal{S}(\alpha_8)$ typically differs for each choice for $\{n_{T_{[0,7]}}\}$. Below we define a notation for the sum property at the moment the first input byte was fed.

Definition 5.5.1. Given the LFSR-stream $a_0 a_1 \ldots$, the sum property value of the cipher's internal state after encrypted input byte $\{b\}$, is fed is $\mathcal{S}_{\{b\}}$.

Here the $\mathcal{S}_{\{b\}}$ is given by

$$\begin{aligned} \mathcal{S}_\epsilon &:= \mathcal{S}(a_{[9,47]}) \\ \mathcal{S}_{\{b\}} &:= \mathcal{S}(a_{[17,55]}) \qquad\qquad \text{where } \{n_{T_{[0,7]}}\} = \{b\} \end{aligned}$$

Furthermore, the set of all possible internal state bits determined by the sum property after input byte $\{b\}$ is fed is $\mathbf{S}_{\{b\}}$. $\mathbf{S}_{\{b\}}$ is given by

$$\begin{aligned} \mathbf{S}_\epsilon &:= \{x \mid x \in \mathbb{F}_2^{39} \text{ and } \mathcal{S}(x) = \mathcal{S}_\epsilon\} \\ \mathbf{S}_{\{b\}} &:= \{x \mid x \in \mathbb{F}_2^{39} \text{ and } \mathcal{S}(x) = \mathcal{S}_{\{b\}}\} \end{aligned}$$

5.5.1 Combining two sum properties

Suppose we have determined $\mathcal{S}_\epsilon$, and also $\mathcal{S}_{\{b\}}$ with high probability, for a certain input byte $\{b\}$. Since every entry $u \in \mathbf{S}_\epsilon$ is a candidate for $a_9 a_{10} \ldots a_{47}$, and every $v \in \mathbf{S}_{\{b\}}$ is a candidate for $a_{17} a_{18} \ldots a_{55}$, every u must have a corresponding v such that $u_8 u_9 \ldots u_{38} = v_0 v_1 \ldots v_{30}$ and vice versa.

This property can be evaluated for the odd and even LFSR state bits separately. This allows us to eliminate candidate keys prior to building the entire candidate list.

Building a candidate list from $\mathcal{S}_\epsilon$ and $\mathcal{S}_{\{b\}}$ is done as follows

(i) We take all pairs $(p, q) \in [0, 16] \times [0, 16]$ for which $\mathcal{S}_\epsilon = p(16 - q) + (16 - p)q$.

(ii) We do the same for time 8: for a certain encrypted input byte $\{b\}$, for which we know $\mathcal{S}_{\{b\}}$ with high probability, we determine all pairs $(r, s) \in [0, 16] \times [0, 16]$ for which $\mathcal{S}_{\{b\}} = r(16 - s) + (16 - r)s$.

(iii) For each pair (p, q), we iterate through all pairs (r, s). For each $x \in T_{O_p}$, we look up all entries $y \in T_{O_r}$ such that $x_4 x_5 \ldots x_{19} = y_0 y_1 \ldots y_{15}$. If none exist, then x is an impossible candidate for $a_9 a_{11} \ldots a_{47}$. Let $z_O \in \mathbb{F}_2^{24}$ be a candidate for $a_9 a_{11} \ldots a_{55}$. It is constructed by taking $z_O := x_0 x_1 x_2 x_3 y$. The even case is similar: for each $x \in T_{E_q}$, we look up all entries $y \in T_{E_s}$ such that $x_4 x_5 \ldots x_{18} = y_0 y_1 \ldots y_{14}$. $z_E \in \mathbb{F}_2^{23}$ is a candidate for $a_{10} a_{12} \ldots a_{54}$ and is constructed by taking $z_E := x_0 x_1 x_2 x_3 y$.

(iv) We now define $z \in \mathbb{F}_2^{47}$, which is a candidate for $a_{[9,55]}$. It is constructed by combining every z_O with every z_E and taking $z := z_{O_0} z_{E_0} z_{O_1} z_{E_1} \ldots z_{O_{22}} z_{E_{22}} z_{O_{23}}$. Stage 3 of Chapter 5 describes how the resulting candidate list is used to perform a key recovery.

Optionally, the size of the search space is determined by, rather than actually constructing the candidate list, multiplying the number of candidates z_O by the number of candidates z_E for each pair (p, q) and (r, s) and summing them up.

We somewhat naively assume that $\mathcal{S}_\epsilon$ and $\mathcal{S}_{\{b\}}$ are statistically independent. In case we are concerned with a random $\{b\}$, we gain a complexity drop of 2.28 bits on average, in addition to the same drop described in Section 5.3. However, in practice, the drop is even greater since relatively few nonces are required for determining a byte $\{b\}$ for which $\mathcal{S}_{\{b\}}$ is an extreme value, i.e. 0 or 256 (yielding a drop of approximately 5.15

bits). The same holds to a lesser extent for values 32 and 224 (approximately 6.81 bits). Moreover, all sum property values except 128 yield a greater complexity drop than 2.28 since their corresponding probabilities are below the average (Figure 5.3.1), hence so are the numbers of corresponding possible cipher states.

The remainder of this chapter is concerned only with extending step (iii) such that we eliminate additional z_O and z_E candidates. The methodology of constructing a candidate list presented here is final.

5.5.2 Combining additional sum properties

Everything presented in this section aims to eliminate odd candidates z_O in step (iii) of the methodology described above, and hence further drop the computational complexity. It is also applicable to even candidates. However, to avoid repetition, we will not concern ourselves with this.

Suppose, in addition to $\mathcal{S}_\epsilon$ and $\mathcal{S}_{\{b\}}$, we also determine $\mathcal{S}_{\{b'\}}$ with high probability, for encrypted input byte $\{b'\}$, where $\{b\} \neq \{b'\}$. We refer to $a_0 a_1 \ldots$ and $a'_0 a'_1 \ldots$ as the LFSR-stream resulting from feeding $\{b\}$ and $\{b'\}$ as input, respectively.

Suppose $\{b\}$ and $\{b'\}$ have a common prefix of i bits, i.e. $\{b_j\} = \{b'_j\}$ for all $j < i$. In step (iii) of the methodology from the previous paragraph we, for each pair (p, q), iterate through all pairs (r, s). Within this iteration, we will now also go through all values $r' \in [0, 16]$, for which a value $\mathcal{S}_{\{b'\}} = r'(16 - k) + (16 - r')k$ exists, where $k \in [0, 16]$. Recall that every $y \in T_{O_r}$ is a candidate for $a_{17} a_{19} \ldots a_{55}$. Per lemma 5.2.2, we know that $a_{48+j} = a'_{48+j}$ for all $j < i$. Therefore, a $y' \in T_{O_{r'}}$ must exist such that $y_j = y'_j$ for all $j < 16 + \lfloor \frac{1}{2} i \rfloor$. If this is not the case, we can eliminate y as a candidate for $a_{17} a_{19} \ldots a_{55}$.

Next we focus on the remainder of $\{b\}$, i.e. the bits beyond the constant prefix of i bits, to further eliminate candidates. We eliminate the entry $y \in T_{O_r}$ if we can prove that the candidate is invalid, *regardless* of what value is stored in the even bits of the LFSR.

We follow Definitions 3.5.1 and 3.5.2 and obtain, for $0 \leq k < 8$

$$\begin{aligned} a_{48+k} =& f(a_{9+k} a_{11+k} \ldots a_{47+k}) \oplus \{b_k\} \oplus u_k \oplus \\ & L(a_{0+k} a_{5+k} \ldots a_{43+k}) \end{aligned}$$

We take $j := i$ (note that later we want to increase j) and take the difference between $\{a_{48+j}\}$ and $\{a'_{48+j}\}$. We get

$$\begin{aligned} a_{48+j} \oplus a'_{48+j} =& f(a_{9+j} a_{11+j} \ldots a_{47+j}) \oplus \{b_j\} \oplus u_j \oplus \\ & L(a_{0+j} a_{5+j} \ldots a_{43+j}) \oplus \\ & f(a'_{9+j} a'_{11+j} \ldots a'_{47+j}) \oplus \{b'_j\} \oplus u_j \oplus \\ & L(a'_{0+j} a'_{5+j} \ldots a'_{43+j}) \end{aligned}$$

Obviously, u_j is XOR-ed twice, so it is canceled out. Recall that we are concerned with odd bits, thus j is odd. Therefore, $f(a_{9+j} a_{11+j} \ldots a_{47+j})$ depends only on even LFSR stream bits. We introduce an invariant stating that the even bits are equal in both states:

$$a_{9+j} a_{11+j} \ldots a_{47+j} = a'_{9+j} a'_{11+j} \ldots a'_{47+j} \tag{5.1}$$

Given that $j = i$, we know that the invariant holds. Hence, also $f(a_{9+j} a_{11+j} \ldots a_{47+j}) = f(a'_{9+j} a'_{11+j} \ldots a'_{47+j})$, regardless of what the actual value for $a_{9+j} a_{11+j} \ldots a_{47+j}$ is. Furthermore, since all even positioned bits fed to the feedback function L are equal, they

are canceled out. Thus, the above is equivalent to

$$a_{48+j} \oplus a'_{48+j} = \{b_j\} \oplus \{b'_j\} \oplus a_{42+j} \oplus a'_{42+j}$$

Hence, in order for candidate y to be valid, a y' must exist such that

$$y_{16+\lfloor \frac{1}{2}j \rfloor} \oplus y'_{16+\lfloor \frac{1}{2}j \rfloor} = \{b_j\} \oplus \{b'_j\} \oplus y_{13+\lfloor \frac{1}{2}j \rfloor} \oplus y'_{13+\lfloor \frac{1}{2}j \rfloor}$$

Suppose such a y' indeed exists such that the above is true. At this point, we need not immediately accept y as a valid candidate. Rather, we may test whether the above also holds for $j \leftarrow j + 2$. However, in order to do so, we must first check whether invariant (5.1) still holds. Given that it holds for j, all we need to do is verify that $a_{49+j} = a'_{49+j}$. Following the definition of a_{49+j}, we obtain

$$\begin{aligned} a_{49+j} \oplus a'_{49+j} = f(a_{10+j}a_{12+j} \dots a_{48+j}) \oplus \{b_{j+1}\} \oplus u_{j+1} \oplus \\ L(a_{1+j}a_{6+j} \dots a_{44+j}) \oplus \\ f(a'_{10+j}a'_{12+j} \dots a'_{48+j}) \oplus \{b'_{j+1}\} \oplus u_{j+1} \oplus \\ L(a'_{1+j}a'_{6+j} \dots a'_{44+j}) \end{aligned}$$

Similar as before, u_{j+1} is XOR-ed twice and hence canceled out. Also the even positioned bits and all odd bits positioned between 0 and $47+i$ fed to the feedback function are equal, thus canceled out. Hence, the above becomes

$$\begin{aligned} a_{49+j} \oplus a'_{49+j} = \{b_{j+1}\} \oplus \{b'_{j+1}\} \oplus f(a_{10+j}a_{12+j} \dots a_{48+j}) \oplus \\ f(a'_{10+j}a'_{12+j} \dots a'_{48+j}) \oplus a_{42+j} \oplus a'_{42+j} \oplus a_{44+j} \oplus a'_{44+j} \end{aligned}$$

Translating this into terms of y and y' again, we obtain (recall x in step (iii) from the methodology described in the last paragraph)

$$\begin{aligned} \{b_{j+1}\} \oplus \{b'_{j+1}\} \ \oplus f(x_{[1+\lfloor \frac{1}{2}j \rfloor, 3]}y_{[0,16+\lfloor \frac{1}{2}j \rfloor]}) \oplus f(x_{[1+\lfloor \frac{1}{2}j \rfloor, 3]}y'_{[0,16+\lfloor \frac{1}{2}j \rfloor]}) \oplus \\ y_{13+\lfloor \frac{1}{2}j \rfloor} \oplus y'_{13+\lfloor \frac{1}{2}j \rfloor} \oplus y_{14+\lfloor \frac{1}{2}j \rfloor} \oplus y'_{14+\lfloor \frac{1}{2}j \rfloor} \end{aligned}$$

If the above evaluates to 0, we have proven that, in case y is valid, then $a_{49+j} = a'_{49+j}$, and hence we have proven that invariant (5.1) still holds. We proceed by attempting to disprove the validity of y once more with $j \leftarrow j + 2$. Otherwise, we stop and accept y as a candidate. In case we reach $j = 7$ we always stop and accept y as a candidate.

In case no y' exists such that y is accepted, we have proven the invalidity of y and eliminate it.

Obviously, the differential analysis presented here can be repeated with other input bytes $\{b'\}$, which will result in the elimination of additional key candidates.

Due to the sheer complexity of analyzing the average size of the leftover complexity yielded by the differential analysis, we will not concern ourselves with this. Practical experiments indicate that it is sensible to assume a drop of approximately 1 bit per byte $\{b'\}$ involved in the analysis. However, it is important to note that a single incorrect guess for $\mathcal{S}_{\{b'\}}$ will likely cause the correct key to be absent from the resulting leftover search space.

In the next section, we present another independent property of the cipher's internal state that we can deduce by observing the ciphertext. We may use this property to eliminate additional key candidates, hence even further dropping the computational complexity.

5.6 Filter Flip Property

The second property that can be observed by analyzing the ciphertext is what we name the filter flip property. It was first documented in the literature by Garcia et al. in 2009

[GRVS09].

Lemma 5.6.1. *Suppose we obtained two encrypted nonces $\{n_T\}$ and $\{n'_T\}$. Their corresponding LFSR-streams are $a_0a_1\ldots$ and $a'_0a'_1\ldots$, respectively and their parity bits are p_i and p'_i for all $i \in \mathbb{N}$. Suppose that we observe that all bytes before byte i, where $i \in [0,3]$, are equal and that only the last bit of byte i differs, i.e. $\{n_{T_{[0,8i+7]}}\} = \{n'_{T_{[0,8i+7]}}\} \oplus 1$. Furthermore, we observe that $\{p_i\} = \{p'_i\}$.*

Then $f(\alpha_{8i+8}) \neq f(\alpha_{8i+8} \oplus 1)$.

Proof. By Lemma 5.2.2, we obtain that $a_{[0,8i+54]} = a'_{[0,8i+54]}$ and $a_{8i+55} \neq a_{8i+55}$. It follows that $\alpha_{8i+8} = \alpha'_{8i+8} \oplus 1$. By the same lemma we also obtain $n_{T_{[0,8i+7]}} = n'_{T_{[0,8i+7]}} \oplus 1$. Trivially, we hence also have $n_{T_{[8i,8i+7]}} = n'_{T_{[8i,8i+7]}} \oplus 1$. Since the plaintext bytes differ by exactly one bit, we get $p_i \neq p'_i$. However, we observed that $\{p_i\} = \{p'_i\}$. Hence

$$ks_{8i+8} \neq ks'_{8i+8}$$

Which is equivalent to

$$f(\alpha_{8i+8}) \neq f(\alpha_{8i+8} \oplus 1)$$

□

Suppose that we find a case of $\{n_{T_{[8i,8i+7]}}\}$ where we observed that the above does not apply, i.e. it is known that $f(\alpha_{8i+7}) = f(\alpha_{8i+7} \oplus 1)$. Then we can evaluate the filter flip property on even bits. The following lemma states this.

Lemma 5.6.2. *Suppose we obtained two encrypted nonces $\{n_T\}$ and $\{n'_T\}$, Their corresponding LSFR-streams are $a_0a_1\ldots$ and $a'_0a'_1\ldots$, respectively and their parity bits are p_i and p'_i for all $i \in \mathbb{N}$. Suppose we observe that all bytes before byte i, where $i \in [0,3]$, are equal and that only the second last bit of byte i differs, i.e. $\{n_{T_{[0,8i+7]}}\} = \{n'_{T_{[0,8i+7]}}\} \oplus 2$. Furthermore, we observe that $\{p_i\} = \{p'_i\}$, and we have determined through Lemma 5.6.1 that $f(\alpha_{8i+8}) = f(\alpha_{8i+8} \oplus 1)$.*

Then $f(\alpha_{8i+7}) \neq f(\alpha_{8i+7} \oplus 1)$.

Proof. By Lemma 5.2.2, we obtain $\alpha_{8i+7} = \alpha'_{8i+7} \oplus 1$. Given is $f(\alpha_{8i+8}) = f(\alpha_{8i+8} \oplus 1)$. We distinct two cases

$f(\alpha_{8i+7}) = f(\alpha_{8i+7} \oplus 1)$:
Since $\{n_{T_{8i+7}}\} = \{n'_{T_{8i+7}}\}$ (given), we obtain $a_{48+8i+7} = a'_{48+8i+7}$ and $n_{T_{8i+7}} = n'_{T_{8i+7}}$. Since $n_{T_{[8i,8i+7]}}$ and $n'_{T_{[8i,8i+7]}}$ differ by exactly one bit, we know that $p_i \neq p'_i$. By definition of f, the value of $a_{48+8i+6}$ is irrelevant for computing ks_{8i+8}. Hence, $ks_{8i+8} = ks'_{8i+8}$. Finally, we determine $\{p_i\} \neq \{p'_i\}$, which is contrary to the observed.

$f(\alpha_{8i+7}) \neq f(\alpha_{8i+7} \oplus 1)$:
Complementary to the above, since $\{n_{T_{8i+7}}\} = \{n'_{T_{8i+7}}\}$ (given), we obtain $a_{48+8i+7} \neq a'_{48+8i+7}$ and $n_{T_{8i+7}} \neq n'_{T_{8i+7}}$. Since $n_{T_{[8i,8i+7]}}$ differ by exactly two bits, we know that $p_i = p'_i$. By definition of f, the value of $a_{48+8i+6}$ is irrelevant for computing ks_{8i+8}. Thus, we obtain $ks_{8i+8} = ks'_{8i+8}$. Since we know that $f(\alpha_{8i+8}) = f(\alpha_{8i+8} \oplus 1)$ (given), we determine $\{p_i\} = \{p'_i\}$.

Hence, when we observe that $\{p_i\} = \{p'_i\}$, we can deduce that $f(\alpha_{8i+7}) \neq f(\alpha_{8i+7} \oplus 1)$. □

We state without proof that the lemma above can be applied as well when $\{n_{T_{[0,8i+7]}}\} = \{n'_{T_{[0,8i+7]}}\} \oplus 3$, i.e. both the second-last and the last bit of byte i differ. In this case, however, the result is flipped, i.e. $f(\alpha_{8i+7}) \neq f(\alpha_{8i+7} \oplus 1)$ if $\{p_i\} \neq \{p'_i\}$.

We continue with the lemma showing that only approximately 9.4% of the possible inputs to the filter function f have this property [GRVS09].

Lemma 5.6.3. *Let $Y_0, \ldots, Y_4$ be independent uniformly distributed random variables over $\mathbb{F}_2$. Then*

$$P[f_b(Y_0, Y_1, Y_2, Y_3) \neq f_b(Y_0, Y_1, Y_2, \overline{Y_3})] = \tfrac{1}{4}$$
$$P[f_c(Y_0, Y_1, Y_2, Y_3, Y_4) \neq f_c(Y_0, Y_1, Y_2, Y_3, \overline{Y_4})] = \tfrac{3}{8}.$$

Proof. By inspection. □

Since only the twenty bits that are input to f are relevant, all states $x \in \mathbb{F}_2^{20}$ such that $f(x) \neq f(x \oplus 1)$ can be easily generated. Below the set of these states F is defined

$$F := \{x \mid x \in \mathbb{F}_2^{20} \text{ and } f(x) \neq f(x \oplus 1)\}$$

We can use the differential analysis described in the previous section to further narrow down the search space. We may do so by applying it to F, rather than T_{O_r} in case we observe that $f(\alpha_8) \neq f(\alpha_8 \oplus 1)$ for a certain input byte $\{b'\}$.

Practical experiments indicate that, for every filter flip property observed, we may assume a complexity drop of approximately $\frac{1}{2}$ bits during the differential analysis described in the previous section.

In the next chapter we shall more concretely analyze the performance of the attack by means of simulations.

Chapter 6

Performance analysis

In this section, we analyze the performance of the attack. We implemented the attack and ran simulations, where we vary the number of nonces gathered and the probability threshold. The performance of the attack is expressed by the size of the resulting leftover search space, which is determined as described in step (iv) in Section 5.5.1. The sum property value and the filter flip property being present both depend on the cipher's internal state. Hence, the resulting complexity depends heavily on the key. Due to this fact, in order to assess the overall efficiency of the attack, we simulated the attack using 100 randomly chosen keys.

Figure 6.0.1 contains a graph depicting the median leftover complexity. The translucent planes depict the second and third quartile. From this figure we can observe that the leftover complexity quickly becomes within reach of solving on ordinary hardware within minutes. Typically, after collecting approximately 10,000 - 20,000 nonces, the leftover complexity is solvable even within seconds. The trade-off between gathering additional nonces or starting a brute force attempt within the leftover search space depends on which is on the upper hand: the nonce-retrieving hardware or the computational power we have at our disposal.

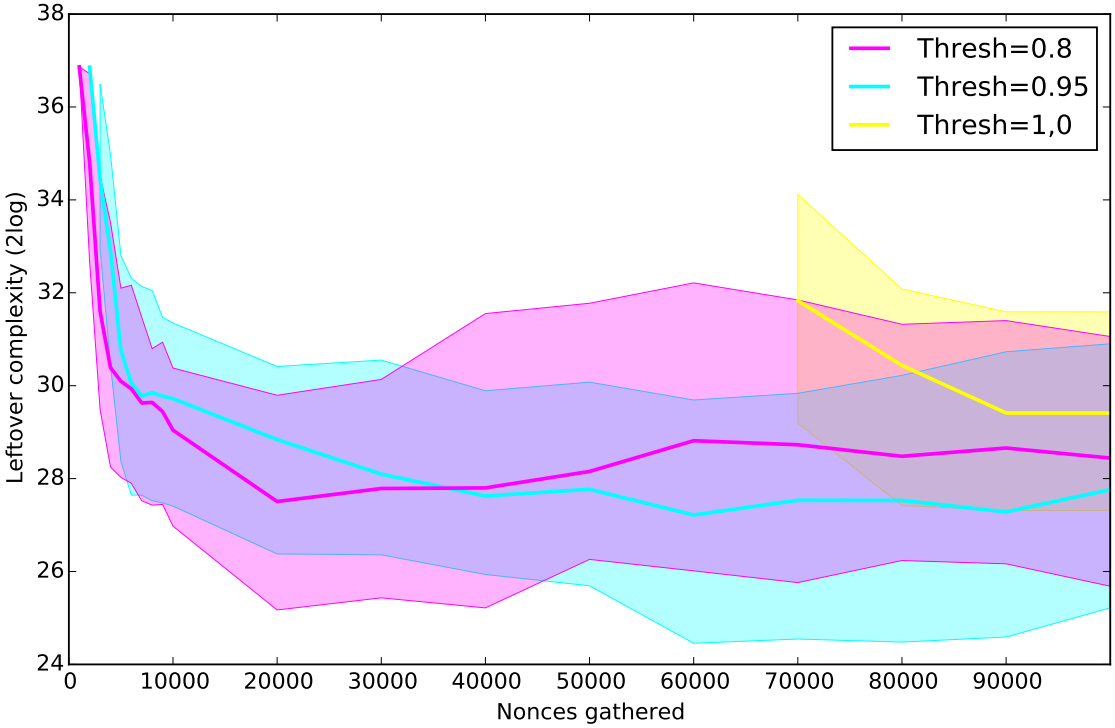


FIGURE 6.0.1: Median leftover complexity

Contrary to the expectations, the leftover complexity may increase slightly when the number of nonces increases. We suspect this is due to the fact that, in our implementation, we select a single byte $\{b\}$ and perform the differential analysis presented in Section 5.5 against all other bytes $\{b'\}$. The selection of $\{b\}$ is based on heuristics which we will not explain in detail. The consequence of this is that the analysis runs significantly faster than when we would perform the analysis for *every* possible $\{b\}$ against all other bytes $\{b'\}$ and subsequently select the smallest resulting set, at the cost of the resulting search space becoming somewhat suboptimal. However, the total time needed to recover a key is decreased.

Every time we choose to involve another input byte/sum property pair in the differential analysis (i.e. the sum property value is known with a probability exceeding the threshold chosen), the leftover search space decreases in size. Since sum property values at time 8 are determined probabilisticly, the probability that the correct keys exists within the leftover search space also decreases. Therefore, another aspect of the performance analysis is determining the actual probability that the correct key is located within the leftover search space, given the probability threshold chosen.

Figure 6.0.2 depicts the probability of the correct key being within the leftover search space. One may expect it to rapidly decrease in case we involve a large number of input bytes in the analysis. Fortunately for the attacker, this is not the case. This is because the sum property values for each input byte are not statistically independent from one another.

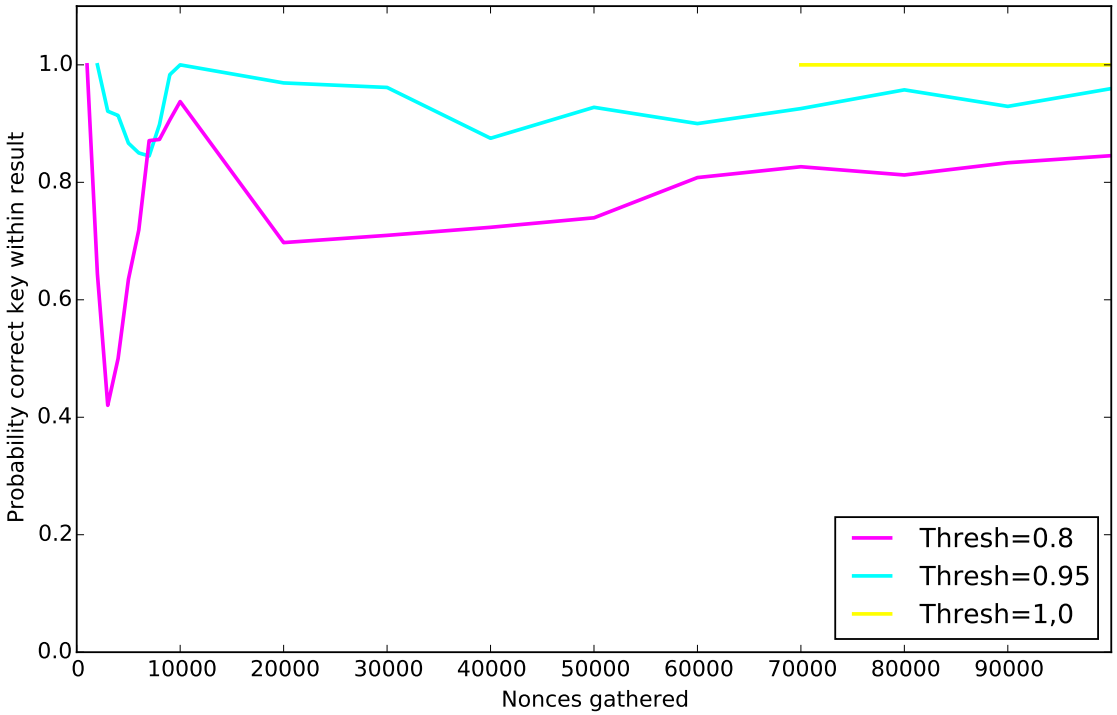


FIGURE 6.0.2: Leftover search space with the correct key.

Finally, we should highlight that our implementation of the attack only returns the leftover search space if at least a single input byte is involved in the analysis, i.e. the probability of guessing the sum property correctly exceeds the chosen threshold for at least a single input byte at time 8). Though this is not strictly necessary, as one could simply take the set of all possible values for the internal state at time 8. However, implementing the attack this way resulted in cleaner code. Therefore, in the statistics

depicted above, every sample wherein we do not assign a sum property value to any of the input bytes is not taken into account.

Figure 6.0.3 depicts the number of samples having at least a single sum property guess exceed the probability threshold. Thus, depicting the number of samples being taken into account in the other simulations presented in this chapter.

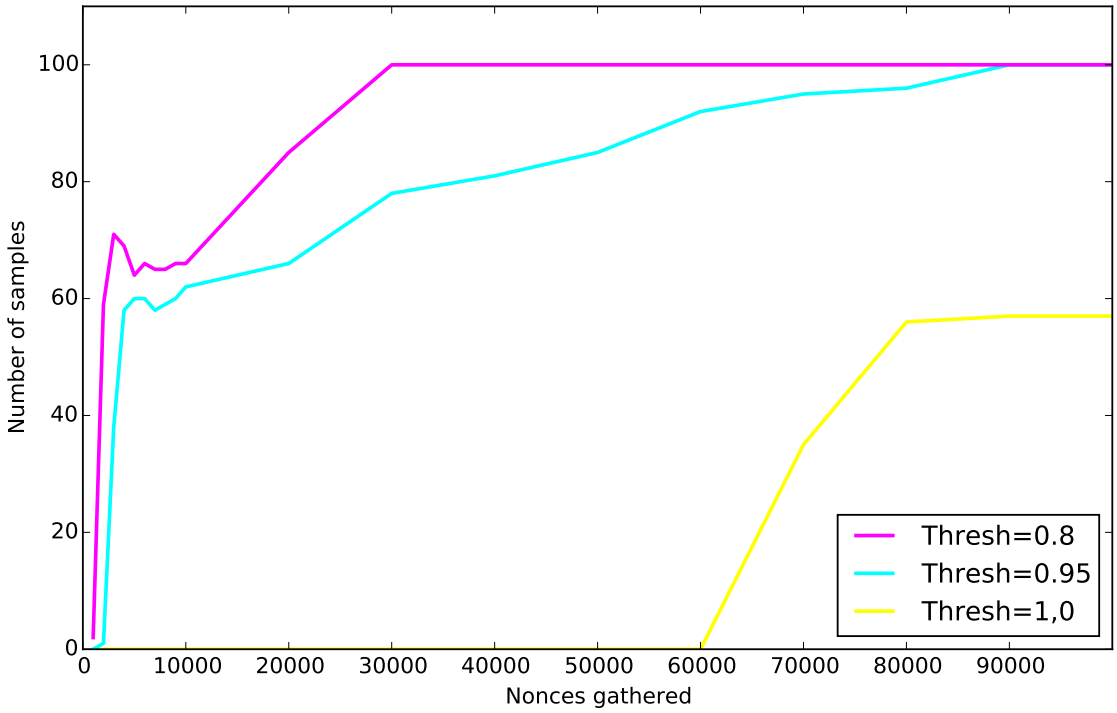


FIGURE 6.0.3: Number of samples involved in the statistics

Chapter 7

Conclusion

Over the last years there are a number of vulnerabilities and attacks identified in the cryptography and implementation of MIFARE Classic cards. The most serious of them are the card-only attacks, which can recover the secret key simply through wireless interaction with a card in an uncontrolled environment. System integrators consider these attacks as one of most serious threats to their MIFARE Classic based systems, since it allows the attacker to avoid camera detection.

We are the first to discover a card-only attack that depends solely on the design issues of the cipher and authentication protocol. To the best of our knowledge, every MIFARE Classic compatible card that is currently in circulation is vulnerable to our attack. Table 7.0.1 shows a comparison between our attack and previous card-only attacks found in the literature.

TABLE 7.0.1: Comparison of card-only attacks

Attack	**Traces**	**Gather**	**Compute**	[a]	[b]
[GRVS09]	2	$<$ 1s	$<$ 1s	×	✓
[Cou09]	300	3m	$<$ 1s	×	×
[CHC+14]	$\sim$ 1,000,000	10-20h	2-15m	✓	×
Our	$\sim$ **10,000**	**6-12m**	**5-10m**	✓	✓

[a] Does not require a weak PRNG
[b] Does not require the error code after a failed authentication

Hardened MIFARE Classic cards (e.g. SmartMX and MIFARE Plus) are not susceptible to previously published card-only attacks. However, they are vulnerable to our attack described in this thesis. Moreover, in order to mitigate our attack, backwards compatibility with the MIFARE Classic protocol is inherently broken. Therefore, we conclude that all MIFARE Classic compatible cards should be regarded as plain memory cards and system integrators can no longer trust their data's authenticity and confidentiality.

We have fully implemented and tested our attacks in practice on various hardened MIFARE Classic cards and recovered secret keys within minutes. Furthermore, we present an extensive complexity analysis with the theoretical boundaries to give a better estimate of the average running-time.

The only prerequisite of our attack is that a single key must be known in advance. However, in practice this requirement is almost always satisfied due to the massive deployment of cards that use a default key for at least one or more memory sectors. Mitigation is typically costly and nontrivial, since these sectors are commonly used to store key diversification information.

We have notified the manufacturer NXP seven months in advance of publication and practically demonstrated our attack on their hardened MIFARE Classic cards. After

notification, we attended several meetings to discuss the attack and its impact. Furthermore, they asked us to review a draft of their customer notification letter wherein they acknowledge our work and discourage to use the MIFARE Classic compatible technology in the future.

7.1 Recommendations

We strongly advice system integrators to migrate away from MIFARE Classic compatible systems and start using strong and cryptographically secure systems. There are many alternative contactless smart cards that support well-studied cryptographic algorithms and formally verified authentication protocols. However, system integrators which are absolutely unable to upgrade their infrastructure could temporarily consider the following palliating countermeasures:

(i) Deploy hardened cards and diversify all keys. – Requires the attacker to perform a different attack prior to ours, such as [GKGM+08], involving either eavesdropping or communication with a reader. This has to take place in a controlled environment, risking camera detection.

(ii) Perform authenticity and integrity checks in the backoffice on a regular basis to detect fraudulent transaction.

Bibliography

[AK03] Frederik Armknecht and Matthias Krause. Algebraic attacks on combiners with memory. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, pages 162–175. Springer-Verlag, 2003.

[And91] Ross J Anderson. Tree functions and cipher systems. *Cryptologia*, 15(3):194–202, 1991.

[And95] Ross Anderson. Searching for the optimum correlation attack. In *2nd International Workshop on Fast Software Encryption (FSE 1994)*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer-Verlag, 1995.

[BDR+96] Matt Blaze, Whitfield Diffie, Ronald L Rivest, Bruce Schneier, and Tsutomu Shimomura. Minimal key lengths for symmetric ciphers to provide adequate commercial security. a report by an ad hoc group of cryptographers and computer scientists. Technical report, DTIC Document, 1996.

[Bih97] Eli Biham. A fast new DES implementation in software. In *4th International Workshop on Fast Software Encryption (FSE 1997)*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer-Verlag, 1997.

[BS00] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology–ASIACRYPT 2000*, pages 1–13. Springer, 2000.

[CCCS92] Paul Camion, Claude Carlet, Pascale Charpin, and Nicolas Sendrier. On correlation-immune functions. In *11th International Cryptology Conference, Advances in Cryptology (CRYPTO 1991)*, volume 576 of *Lecture Notes in Computer Science*, pages 86–100. Springer-Verlag, 1992.

[CHC+14] Yi-Hao Chiu, Wei-Chih Hong, Li-Ping Chou, Jintai Ding, Bo-Yin Yang, and Chen-Mou Cheng. A Practical Attack on Patched MIFARE Classic. In *Information Security and Cryptology*, pages 150–164. Springer, 2014.

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In *21st International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2002)*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer-Verlag, 2002.

[CM03] Nicolas T Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *22nd International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2003)*, pages 345–359. Springer-Verlag, 2003.

[Cou09] Nicolas T Courtois. The dark side of security by obscurity and cloning Mifare Classic rail and building passes, anywhere, anytime. *SECRYPT: International Conference on Security and Cryptography*, 2009.

[CP02] Nicolas T Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *8th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2002)*, pages 267–287. Springer-Verlag, 2002.

[CS91] Vladimir Chepyzhov and Ben Smeets. On a fast correlation attack on certain stream ciphers. In *10th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1991)*, volume 547 of *Lecture Notes in Computer Science*, pages 176–185. Springer-Verlag, 1991.

[DHW+12] Benedikt Driessen, Ralf Hund, Carsten Willems, Carsten Paar, and Thorsten Holz. Don't trust satellite phones: A security analysis of two satphone standards. In *33rd IEEE Symposium on Security and Privacy (S&P 2012)*, pages 128–142. IEEE, 2012.

[FGT92] Philippe Flajolet, Daniele Gardy, and Loÿs Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3):207–229, 1992.

[FJ03] Jean-Charles Faugere and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, pages 44–60. Springer-Verlag, 2003.

[GKGM+08] Flavio D Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter Van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE classic. In *Computer Security-ESORICS 2008*, pages 97–114. Springer, 2008.

[GKGV12] Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Tutorial: Proxmark, the swiss army knife for RFID security research. Technical report, Radboud University Nijmegen, 2012.

[Gol96] Jovan Dj Golić. On the security of nonlinear filter generators. In *3rd International Workshop on Fast Software Encryption (FSE 1996)*, volume 1039 of *Lecture Notes in Computer Science*, pages 173–188. Springer-Verlag, 1996.

[Gol97] Jovan Dj. Golić. Cryptanalysis of alleged A5 stream cipher. In *16th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1997)*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, 1997.

[GRVS09] 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.

[Hel80] Martin E Hellman. A cryptanalytic time-memory trade-off. *Information Theory, IEEE Transactions on*, 26(4):401–406, 1980.

[Hil29] Lester S. Hill. Cryptography in an algebraic alphabet. *American Mathematical Monthly*, 36(6):306–312, 1929.

[ISO99] Mechanisms using symmetric encipherment algorithms (ISO/IEC 9798 part 2), 1999. International Organization for Standardization (ISO).

[ISO01] Identification cards — contactless integrated circuit cards — proximity cards (ISO/IEC 14443), 2001.

[JJ00] Thomas Johansson and Fredrik Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In *20th International Cryptology Conference, Advances in Cryptology (CRYPTO 2000)*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315. Springer-Verlag, 2000.

[JS97] Norman D. Jorstad and Landgrave T. Smith. Cryptographic algorithm metrics. In *20th National Information Systems Security Conference*. National Institute of Standards and Technology (NIST), 1997.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9(1):5–38, 1883.

[KGHG08] Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D Garcia. A practical attack on the MIFARE Classic. In *Smart Card Research and Advanced Applications*, pages 267–282. Springer, 2008.

[Kuh88] GJ Kuhn. Algorithms for self-synchronizing ciphers. In *1st Southern African Conference on Communications and Signal Processing (COMSIG 1988)*, pages 159–164. IEEE, 1988.

[MAD07] Mifare application directory. http://www.nxp.com/acrobat_download/other/identification/M001830.pdf, May 2007.

[Mar57] Harry M Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3(3):255–269, 1957.

[MS88] Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers. In *7th Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1988)*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer-Verlag, 1988.

[Mul56] David E Muller. A method for solving algebraic equations using an automatic computer. *Mathematical Tables and Other Aids to Computation*, 10(56):208–215, 1956.

[MV15] Carlo Meijer and Roel Verdult. Ciphertext-only cryptanalysis on hardened mifare classic cards. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 18–30. ACM, 2015.

[NESP08] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, volume 28, 2008.

[NP07] Karsten Nohl and Henryk Plötz. Mifare, little security, despite obscurity. In *24th congress of the Chaos Computer Club in Berlin*, 2007.

[PHI98] MIFARE Classic 1k, MF1ICS50. Public product data sheet, July 1998. Philips Semiconductors.

[Sie84] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–780, 1984.

[Sie85] Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, 100(1):81–85, 1985.

[SN97] National Institute for Standards and Technology (NIST). Announcing request for candidate algorithm nominations for the advanced encryption standard (AES). *Federal Register*, 62(177):48051–48058, 1997.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[TT80] Moiez A. Tapia and Jerry H. Tucker. Complete solution of boolean equations. *IEEE Transactions on Computers*, 100(7):662–665, 1980.

[Ver15] Roel Verdult. *The (in)security of proprietary cryptography*. PhD thesis, Radboud University, The Netherlands and KU Leuven, Belgium, April 2015.

[VGB12] Roel Verdult, Flavio D. Garcia, and Josep Balasch. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium (USENIX Security 2012)*, pages 237–252. USENIX Association, 2012.

[VKGG12] Roel Verdult, Gerhard de Koning Gans, and Flavio D. Garcia. A toolbox for RFID protocol analysis. In *4th International EURASIP Workshop on RFID Technology (EURASIP RFID 2012)*, pages 27–34. IEEE Computer Society, 2012.