THE DARK SIDE OF SECURITY BY OBSCURITY
and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime

Nicolas T. Courtois
University College London, Computer Science, Gower street, WC1E 6BT, London, UK

Keywords: Access control, RFID, contactless smart cards, MiFare Classic, London Oyster card, OV-Chipcard, trade secrets, secure hardware devices, reverse-engineering, electronic subversion, covert channels, implementation backdoors, critical application development management, information assurance, crime science.

Abstract: MiFare Classic is the most popular contactless smart card with some 200 millions copies in circulation worldwide. At Esorics 2008 Dutch researchers showed that the underlying cipher Crypto-1 can be cracked in as little as 0.1 seconds if the attacker can eavesdrop the RF communications with the (genuine) reader.
We discovered that a MiFare classic card can be cloned in a much more practical card-only scenario, where the attacker only needs to be in the proximity of the card for a number of minutes, therefore making usurpation of identity through pass cloning feasible at any moment and under any circumstances. For example, anybody sitting next to the victim on a train or on a plane is now be able to clone his/her pass. Other researchers have also (independently from us) discovered this vulnerability (Garcia et al., 2009). In addition, we discovered that a yet unknown proportion of MiFare Classic cards are even weaker, and we have in our possession a MiFare classic card from a large Eastern-European city that can be cloned in seconds.
The main security vulnerability that we need to address with regard to MiFare Classic is **not** about cryptography, RFID protocols and software vulnerabilities. It is a systemic one: we need to understand how much our economy is vulnerable to sophisticated forms of electronic subversion where potentially one smart card developer can intentionally (or not), but quite easily in fact, compromise the security of of governments, businesses and financial institutions worldwide.

1 INTRODUCTION

The MiFare classic contact-less smart card is the most popular RFID card that is used to protect access to buildings worldwide and in public transportation. For more than 10 years, due to the secrecy of the specification, nobody was able to realize how weak these products actually are. When in 2008, two teams of researchers (Nohl et al., 2008; de Koning Gans et al., 2008) have more or less independently reverse engineered this card and discovered **several** very serious vulnerabilities: an incredibly weak cipher broken in 0.1 s, two extremely weak Random Number Generators (RNG), both in the card and in the readers, and the resulting possibility to reproduce the same keystream. In all key recovery attacks proposed so far the attacker must somewhat "approach" the victim, or be able to install an RF interception equipment inside a building. Can a system that is so badly compromised be shown to be even more insecure? The answer is yes, and we discovered yet another very serious vulnerability of MiFare Classic cards. This vulnerability alone (we don't need to combine it with other known vulnerabilities) allows to recover the key and thus clone cards in the weakest and the most realistic attack scenario yet considered: where the attacker has an occasional (and wireless) access to the victim's card, anywhere, anytime. The attack itself is not a very sophisticated one. It does not exploit any cryptographic vulnerability. The key point is that the product contains a very nasty implementation bug (or maybe rather a backdoor) that allows the attacker that is aware of it to quickly recover the 48-bit key by brute force.

This paper is not that much about cryptography. It is about security of smart card and access control systems in the real life. It seems that there is very little research on some major questions here. What is the role of secrecy in developing secure products in the real life? We will argument that total disclosure (cf. Kerckhoffs' principle in its naive interpretation) is actually rather wrong and counter-productive in the world of smart cards. But then we must realize that the secrecy of a product specification can also be an offensive weapon, with a threat of large scale electronic subversion. We need to have the courage to examine these questions and stop pretending that research in security is about discovering vulnerabilities that are always not intentional. Smart card developers are also potential attackers for all such systems and trade/industrial secrets should always be regarded as a potentially very major security breach. At the occasion we revisit the questions of information assurance and secure smart card development management

2 SECURITY OF SMART CARDS

Security solutions with smart cards are very much unlike open networked systems. Hardware security allows to define clear security boundaries that cannot be breached by, for example, malicious software (but can be breached by technically sophisticated human attackers nevertheless).

2.1 Splitting the Security Perimeter

The usual security boundaries in smart cards are the separation between certain security features that are implemented in hardware (such as Write-Once memory, WORM), the card OS burned in ROM, the applications stored in the Non-Volatile Memory (NVM). On the top of it one can have well isolated Java applets with limited capabilities. These boundaries make it much harder to hack smart cards, also with regard to internal threats (e.g. corrupted developers). In (Schneier and Shostack, 1999), we are warned however that these boundaries are also a source of problems and new totally unsuspected vulnerabilities.

2.2 Hardware Security as a Threat

We need to consider the dark, offensive side of smart cards. For example, the manufacturer of the card can engage in so called 'kleptographic attacks' (Young and Yung, 1996) leaking the keys to the attacker in ways that are more or less impossible to detect even to the financial institution itself that is the card issuer and owner. This can be seen as a form of perfect crime that will maybe never be discovered (nobody have discovered any of the very serious flaws in MiFare classic for more than 10 years). And if discovered, it maybe be impossible to prove the malicious intention, and very hard to establish whether the exploit was actually sold or independently discovered.

Much more frequently, there is an abundant track record of more less innocent mistakes or bugs, that create exploitable vulnerabilities in commercial products. And again it is very hard to know which ones of these may be intentional (is this a bug or a feature?). For example, when MiFare Classic was reverse engineered (Nohl et al., 2008), researchers did NOT immediately realize how weak it was. Neither probably did the developers that many years ago implemented the cipher in hardware. In fact [Karsten Nohl, private communication] though the cipher uses only 3 different Boolean functions, it may seem that there are 6 different Boolean functions. The same Boolean function is used several times and implemented in hardware in a different way. This is a curious and certainly not the most cost-effective implementation strategy. In addition, we have the very unusual regular structure of the LFSR taps that are used by the non-linear Boolean function, see (Garcia et al., 2008). The combination of these two properties, makes that outputs of many Boolean functions inside the cipher are simply exactly equal to outputs of other (seemingly different but in fact identical) Boolean functions used a few clocks earlier or later. This simply means that a large number of logical gates during the hardware computation of Crypto-1 are wasted. Identical values are computed several times. This seems to contradict the idea that Crypto-1 is weak because it was designed by amateurs, or that it is weak to make the chip as inexpensive as possible. One can rather get the impression, that this cipher was rather carefully designed to look much more secure than it actually is, maybe hoping that nobody would notice. We note that the probability that out of 5 Boolean functions on 4 bits, there are only 2 distinct Boolean functions is negligible, about $2^{-3 \cdot 2^4} = 2^{-48}$.

In (Schneier and Shostack, 1999) two key recommandations are given to improve the security of smart cards: more transparency and "placing the user interface under the control of the user". This is very hard to achieve with smart cards with wireless interface such as MiFare Classic, and where the specification was kept secret for so many years. Moreover, as we will explain later, total transparency can also be counter-productive. It appears that the question of what is the best method to develop secure smart card products is a complex and convoluted one.

2.3 Secure Hardware Development Management

In the reference "Smart Card Handbook" (Rankl and Effing, 2003), on page 518 we read: [In smart cards] one design criterion is [..] that absolutely no undocumented mechanisms or functions must be present in the chip ('that's not a bug, that's a feature'). Since they are not documented, they can be unintentionally overlooked during the hardware evaluation and possibly be used later for attacks. The use of such undocumented features is thus strictly prohibited [...]

We learn that the typical situation in the industry is to test final products in a "black box" model, and that test suites do typically include scanning for hidden [debugging] commands that should NOT be left available neither in the hard-mask (ROM) nor the soft-mask (NVM) of the card. The industry also applies the principles of partial secrecy ("need-to-know"), segregation of duties (never one developer should work alone on an application), and monitoring. For example in Common Criteria evaluations (ISO 15408) of smart card, the entire source code may be inspected by an independent company: a government agency or an evaluation lab, preferably mandated and paid by the customer (to avoid conflicts of interests).

Unhappily, not every vulnerability will be found, and we have discovered a very serious vulnerability in MiFare Classic cards.

3 DISCLOSE OR NOT TO DISCLOSE?

The question whether it is ethical to actively research and whether one should disclose security vulnerabilities is not obvious to answer (Rescorla, 2004). Assuming that the researcher is not going to sell the exploit to criminals, the simple fact of publishing it, can have serious consequences. For example, NXP issued a statement (NXP-statement, 2008) regarding the recent attacks (Garcia et al., 2008), saying that publishing the vulnerabilities of MiFare classic will harm system in the field, facilitate "illegal activities" and that upgrades will unhappily take a number of years.

In the security/research community however, a great majority of people (Schneier and Shostack, 1999) will agree that "that the best way to ensure the security of a system is to allow widespread public examination of it". And in (Schneier, 2008) we read that "vulnerability research is vital because it trains our next generation of computer security experts".

It is very naive to believe that disclosing facts about MiFare would not do any harm. It is most likely doing further harm. Even if some criminals have discovered various attacks on MiFare before, some other criminals or terrorists will just now discover new opportunities. However, we also need to look at the harm that comes from non-disclosing. The industry will continue to consider that the security is not important and as a result everybody will be worse-off in the long run.

3.1 Kerckhoff's Principle in Cryptology

More specifically, what about the secrecy of cryptographic algorithms in smart cards, that in many cases are the main and the only "anti-clone" functionality of these products? Most researchers in cryptography contend that the design of cryptographic schemes **must** be public. But in fact this is neither correct nor reasonable. The famous 19th century Dutch cryptologist Auguste Kerckhoffs (Kerckhoffs, 1883) does not recommend full disclosure. He only proposes the design of a system should not require secrecy. When the enemy gets hold of the specification of the system, the security should still remain very good, based on the secret key. Every designer should assume that the cipher is known to the attacker, and it should remain secure also in this case, but this does not entail an obligation to automatically make every cipher public. Modern security is about layering the defenses. If secrecy of the algorithm keeps the attacker at bay for an extra 6 months, it is worth having. But this should not conceal lousy security that will collapse on the very day the specification is disclosed.

In some industries algorithm secrecy is indispensable. For example it very hard and costly to protect smart cards against side channel attacks. Therefore the secrecy of the algorithm is an important asset that really improves the security. For example Pay TV systems have always greatly depended on the secrecy of the embedded algorithms. It is totally unreasonable to ask companies that embed their algorithms in inexpensive hardware that is in the hands of the potential hackers to disclose all their security features The security will be greatly degraded if they do.

Of course, secrecy is a good idea only if these algorithms are good in the first place. Otherwise we are creating an illusion of security which can be as bad or worse than having no security at all.

3.2 Benefits of Disclosure

The main benefit of disclosure is that "the security of the cipher is not in the design, it is in the analysis" [attributed to Schneier]. A cipher that has been

under intense scrutiny over a number of years and yet remains unbroken, will be the most secure one. This is best explained by Karl Popper's philosophy of science. Scientific statements should be hold as provisionally true until proven false. The more a statement withstands attempts to falsify it, the more value it has. Some ciphers such as triple-DES have undergone a Darwinian natural selection process. However other ciphers massively used in the industry such as KeeLoq or MiFare Classic Crypto-1 cipher are just terribly weak (Garcia et al., 2008).

3.2.1 Markets for Security

It appears that markets for security, and for security products tend to be dysfunctional and fail to deliver anything near the most basic level of security. Several issues lie at the roots of this problem. In the computer and IT industry, there are neither legal obligations nor really strong market incentives for the industry to care about security and implement it. There is an asymmetry of information about the security of products. The disclosure of vulnerabilities is then beneficial because it can potentially restore the balance and provide incentives to fix problems. But it does rarely fix the problem, because we cannot change the fact that whatever people do, the security is rather inherently difficult to get right.

In fact the security is mainly a question of **public interest** and private incentives are weak. Insecurity is everywhere, and is a form of nuisance that "pollutes" and degrades our life 'environment'. It frequently affects the customer and the third parties that have little or no choice, and can do nothing about it in the short run, while the people that can fix the problem see no compelling reason of doing so.

It is important to see, that quite frequently, the industry sees these forms of nuisance to the customer as beneficial. It allows them to sell upgrades, renew their product range, and drives faster adoption of new innovative products and services. We need in fact to recognize that insecurity can indeed be beneficial and generate positive outcomes. However very few companies are willing to recognize and address the full scale of the problems that are in fact a direct result of their activity. It is not only a nuisance, but frequently fraud, crime, disruption to other people's lives and third party business processes. The industry behaves rationally, and shifts the real economic cost of their activity on the customers and third parties, and retains the profits of it. But the society has to defend against this practice that often becomes excessive and hurts everybody in the long run.

The problem is really the same as with serious pollution and crime (or fire safety). Corporations that, acting in seemingly innocent self-interest produce insecurity can be as dangerous to the society as criminals that will engulf into the breach and exploit the vulnerabilities. The harm can be as important, regardless whether these vulnerabilities are inadvertent or due to extreme negligence. There is a need for checks and balances and to set up better standards for security. The researchers are one of the very few people that by pinpointing the lousy security of MiFare, defend the public interest. This is a strong argument for allowing the researchers to exercice their freedom of speech.

3.2.2 Discovering the Scale of the Problem

Maybe the main argument for disclosure comes really from a spectacular nature of insecurity of this NXP product. Discovering a real-life exploitable vulnerability of this order of magnitude and yet keeping the specification of the product secret would be really dangerous. Importantly, if there is no sanction against NXP, that can be for example a sanction of the market that will prefer to order smart cards from their competitors, we will all lose. The point is about market economy is that it should promote good technology and good products, and the companies that deliver lousy products and mislead their customers about their security suffer some sort of sanction.

4 CLONING MIFARE CLASSIC

Consider the typical transaction flow in MiFare Classic, following (de Koning Gans et al., 2008; Garcia et al., 2008) and using the same notations:

1. First the reader and the card engage in the anti-collision protocol where the reader learns the unique ID of the card and selects the card.
2. The reader issues a command '60 0X' by which it starts the mutual symmetric-key authentication process between the card and the reader, with the key pertaining to the block number 0X.
3. The card answers with a random n_T on 4 bytes,
4. The reader sends a cryptogram on 8 bytes which is $n_R \oplus ks1$ and $suc^2(n_T) \oplus ks2$.
5. The card responds with 4 bytes, $suc^3(n_T) \oplus ks3$.
6. Then all subsequent communications and data are encrypted and the card will now accept read, write and increment commands for block 0X.

Here n_R is the 32-bit nonce chosen by the reader, *suc* is a certain bijective function, and $(ks1, ks2, ks3)$ are the 96 bits of the keystream produced by the

Crypto-1 stream cipher in a certain somewhat special mode described in (de Koning Gans et al., 2008) after being initialized with n_T and n_R.

4.1 Nice Properties of This Protocol

We can observe that this protocol is designed to make card-only attacks very hard if not totally **impossible**, this including brute force attacks (!) (yet later we will explain that a card-only attack is nevertheless possible, due to a hidden "backdoor" property).

How is this possible? We see that the card never ever answers anything that is related to the secret key before actually the terminal proves the knowledge of this secret key with a 8-byte cryptogram, where the n_R is freely chosen by the reader and clearly the probability that a (false) reader can produce a valid answer is 2^{-32}. This protects **against brute force attacks**: even if the attacker guesses the key, to confirm this key (or reject the wrong one) he needs to query the card once. Each query allows to reject 2^{48-32} keys for which this 8-byte cryptogram $(n_R \oplus ks1, suc^2(n_T) \oplus ks2)$ is valid. In order to perform a brute force attack we need about 2^{47} computations **and** about 2^{32} queries to the card. However, since each transaction with the card takes 0.5 s, the brute force attack requires to query the card for 2^{31} seconds which is 93 years. The brute force attack is infeasible.

4.2 Key Vulnerability

Our key discovery is that for a typical MiFare Classic card (we have tried 20 different cards from different countries, they all had this vulnerability) we have the following:

Notation: We denote by $c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7$ the 8-byte cryptogram sent the card as in point 4, independently whether it comes from a genuine transaction, or from an attempt to spoof the card with a false cryptogram.

Fact 4.2.1 *If we just run the process described above to authenticate the card for which we do* **NOT** *know the key, sometimes, at point 5., and with probability of about 1/256, the card will* **nevertheless respond** *with 4 bits (instead of 4 bytes). These 4 bits are NACK (0x5) that will be encrypted with the next 4 bits of the keystream ks*3.

Explanation: This is a bug, maybe Phlips/NXP can give some explanation. We have found none.

Thus we see that the card is leaking 4 bits of information about the secret key. Moreover, not without considerable effort and imagination, we were able to guess and confirm by computer simulations when exactly this event occurs:

Fact 4.2.2 *The card answers with a 4-bit encrypted NACK if and only if the parity bit for each of the 8 bytes c_i after decryption by the card at point 4, is equal to the next bit of keystream.*

And since, in our spoof attempt, we do naturally set all the parity bits for 8 bytes correctly (every 9th bit transmitted), as when transmitting non-encrypted 8-bytes, this allows to write an extra 8 linear equations on the keystream, where the XOR of all bits in each byte is equal to the next bit of the keystream (in the case of the last byte of ks2, we need to look at the first bit of ks3 transmitted). Thus we get typically 12 bits of information on the keystream each time the card answers[1].

4.3 Our First Card-Only Attack

We need 4 such answers to determine the key by brute force, which for a 48-bit key can be handled in minutes with an FPGA implementation.

How many queries we need to get 4 answers? To get one answer we need about 128 queries on average. Then we do the following: we query the card with the same n_T and the same 24-bit prefix in the 8-byte cryptogram sent to the card in Step 4, and we change only the 4th byte sent. Then it is possible to see that the card responds with probability 1/32 instead of 1/256. We need 3 times 16 steps on average. Overall on average we need $128 + 3 \cdot 16$ queries to the card, which takes about $176 \cdot 0.5$ seconds which is less than 2 minutes. The attacker needs to query the card for 2 minutes before he can clone it by brute force.

4.4 Some Cards are Much Weaker

We have found that for certain MiFare Classic 1 K cards, for example those used in the mass transit system in Kiev, Ukraine the card replays NACK with probability 1 instead of about 1/256. For these cards our attack requires only $12 \cdot 0.5$ seconds to handle, with 4 bits of information revealed per each query. In order to read the full content of the card instantly, this attack will require a substantially higher cost for the hardware to handle the brute force part of the attack (but this can be done by a remote facility accessed through the mobile network).

We would like to stress that to the best our knowledge, these cards are otherwise **indistinguishable** from any other card we have seen: they do give the same ATR (Answer to Reset) equal to "3B8F8001804F0CA000000306030001000000006A" and functionally behave EXACTLY the same.

[1] We could also, but we don't exploit the information contained in all the cases when the card did not reply.

4.5 How to Verify our Claims

What we describe here can be verified by anyone with the open-source smart card reader OpenPCD and the free librfid library, see (Nohl et al., 2008). Another method (recommended) is to use the Texas Instruments TRF7960EVM reader and a modified firmware that can be found on the Internet (Nohl, 2008). With the second method it is quite easy to verify what we say. Basically, in the Python program available to download from (Nohl, 2008), if we just run it to authenticate the card for which we do NOT know the key, it does at certain moments report an error and says: "0 bytes and 4 bits received". These 4 bits are precisely the "backdoor" property we have discovered. In order to read these bits, one needs to modify the software in (Nohl, 2008) and try about 256 times for a card for which we know the key and can recompute $ks3$. The C source code of Crypto-1 is also found inside the code available at (Nohl, 2008).

5 CRYPTOGRAPHIC CARD-ONLY ATTACKS

So far the attack is done by brute force. Can we propose a cryptographic shortcut attack, much faster than brute force? Yes we can.

5.1 Pre-requisite: Timing Attack on the Card Nonce

A pre-requisite for this attack is a strict control of the timing of the attack. By doing so, following a well known vulnerability of MiFare Classic cards (de Koning Gans et al., 2008; Nohl et al., 2008), the attacker can reproduce the same random of the card n_T with high probability.

This can be attempted with the OpenPCD reader in software, but the results are quite poor, typically we could get the most frequent n_T to repeat after about 2^6 queries (instead of 2^{16} that would be obtained with random timing).

Better results can be achieved if we manipulate the smart card reader at the firmware level. In (de Koning Gans et al., 2008) we read that the accuracy of selecting a chosen n_T by controlling the timing with their Proxmark 3 device is about be 1 out of 10. In contrast in Section 2 of (Nohl et al., 2008) we read that with a custom version of OpenPCD firmware, they can reproduce exactly the same n_T each time with near certainty. We have been able to achieve a similar result where we can repeat the same n_T with probability of about 75 % with a modified firmware for TI TRF7960EVM reader developed by Bingsheng Zhang at University College London, similar to the firmware code found in (Nohl, 2008).

5.2 On the Small Variability of the Keystream in Crypto-1

We consider now the Crypto-1 cipher as it is used in decrypting the 8-byte cryptogram that is sent to the card in Step 4. of our transaction flow. The question is how much the keystream $ks1$ that is produced during the decryption of this 8-byte random, (genuine or fake), depends on the actual bits of this random. For example we fix the first 3 bytes and vary only the 4th byte c_3. The answer is that it depends surprisingly little of $ks3$.

Fact 5.2.1 *The probability that the 8 bits of the keystream generated during the decryption of the 4th byte c_3 do NOT depend on this byte is very high, about* $1/6.4$.

Explanation: The probability is surprisingly high, as shown by our computer simulations. This is due to the **very** bad properties of the Boolean functions used in the Crypto-1 cipher, namely to the fact that very frequently the output bit does NOT depend on many input bits. For example, the reader can verify that:

- With probability $10/16$ the output of the combined Boolean function on 20 bits does not depend on the last 4 bits.
- In addition, with probability 3/4 the sub-function that deals with the last (and the latest) 4 bits, does not change if we flip the last bit, and with probability 1/2, it does not change when we flip the second last bit,
- and with probability 1/2, it is always 1 whatever is the change to the last 2 bits.

These are more VERY VERY strong properties that arise with very small probability for random Boolean functions, and constitute **extra evidence in trying to see whether the weakness of MiFare Classic could have occurred as a result of many unhappy random events, or whether this level of insecurity could be considered as intentional.** More research is needed on this topic.

5.3 Impact of Small Variability of ks1

Let us fix the card nonce n_T and a 24-bit prefix (c_0, c_1, c_2) of the 8-byte cryptogram the first half of which is (in a legitimate transaction only) expected to be equal to $n_R \oplus ks1$. Let us assume that the card

answers as in Fact 4.2.1 for some spoof attempt with these 3 bytes. This means that the first 3 our of 8 parity checks give 0.

Now, let us look what happens when we vary the next 4-th byte c_3 of the cryptogram. Actually 9 bits are of concern to us: next bit of ks1 generated (24th since the beginning) that is a constant once (c_0, c_1, c_2) are fixed, and the 8 bits generated and depending on c_3. From Fact 5.2.1 we see that all these 9 bits will be constant with probability $1/6.4$.

However half of this time, and with probability $1/12.8$, their XOR will be 1 and, (cf. Fact 4.2.2) the card will never answer to any of our spoof attempts.

In contrast the other half of thee time, and also with probability $1/12.8$, their XOR will be 0 and, whatever is the byte c_3, the first 4 our of 8 parity checks will give 0, and therefore the card will respond to a spoof attempt in which we only vary the 4th byte c_3 with probability $2^{-(8-4)} = 1/16$. To summarize:

Fact 5.3.1 *If we fix the card nonce n_T and a 24-bit prefix (c_0, c_1, c_2) cryptogram for which the card have already answered, with probability about $1/12.8$ over (n_T, c_0, c_1, c_2) the 8 bits of the keystream generated directly after c_3 is decrypted does not depend on c_3, and the card will answer every a spoof attempt with a random c_3 with probability $1/16$.*

The next question is how to exploit this in a cryptographic attack.

5.4 A Conditional Multiple Differential Property

Further on, by a careful study of the specification of the Crypto-1 cipher as used in the MiFare Classic authentication protocol, see (Garcia et al., 2008), we have observed that in the treatment of this 8-byte cryptogram c_3, (that, in a legitimate transaction only, would be the 4th byte of $n_R \oplus ks1$), the new 8 bits of the state (the main 48-bit n-LFSR in Crypto-1) created at this moment, can all be expressed as a multivariate linear function of previous state bits, and of these 8 "immuable" keystream bits that are just additional 8 variables that we can call $ks1_{24}, \ldots, ks1_{31}$. Moreover, due to the linearity, these 8 variables $ks1_{24}, \ldots, ks1_{31}$ can be eliminated easily: in fact in any pair of executions that differ only in c_3, the differences of state bits in the future will depend ONLY on the difference between the two values of c_3 **and nothing else**. Thus we get:

Fact 5.4.1 *If we assume that the Crypto-1 keystream generated during the decryption of the 4th byte c_3 does NOT depend on this byte, which happens with probability about $1/12.8$, cf. Fact5.3.1, then the difference of the state of the cipher at any moment during the computation of ks2 and ks3 is a fixed multivariate linear function that depends on the differences in c_3 and nothing else.*

Thus we can precompute and store in a table these differences for **ANY** pair of states. We give here directly the (incomplete) result of our precomputation, where we present the actual difference in the future state bits (precisely those used to compute keystream used to encrypt NACK) for certain differences in the first 4-bytes of the 8-byte cryptogram c_3.

```
00000001 8DC1B21F6E10
00000002 1B83643EDC20
00000004 3706C87DB840
```

This operation is linear and could be described by a matrix 8x52 that is fixed and depends only on the LFSR connections of the Crypto-1 cipher.

5.5 Our Shortcut Card-Only Cloning Attack with No Precomputation

From the NXP web page (NXP-web, 2008) we learn some information about a new attack that will be published in (Garcia et al., 2009):

"Card only attacks are possible in lab environments and at considerable precalculation time. This is expected to further evolve into an attack that does not need lab conditions and may require less precalculation time."

Therefore it is not clear whether an efficient attack without a costly pre-computation exists. In this paper we show an attack in this category that is very fast and requires no pre-computation whatsoever. At the moment of writing we ignored what kind of attack is described and studied in (Garcia et al., 2009), and whether it is a purely cryptographic shortcut attack with chosen (or fixed) nonces as ours is, or if it also explores other additional vulnerabilities of the system to get a similar (or different, maybe better) result.

Here is a short description of our attack. In fact some 20 other attacks on MiFare Classic are known to us, however this one is particularly efficient, and yet remains relatively simple to describe.

1. We send 128 queries (in expectancy) with a fixed or random n_T, and a random 8-byte cryptogram at point 4 in the transaction flow (cf. Section 4) to get one case where the card answers with 4 bits.
2. Then we keep the same fixed timing now, so that the n_T of the card will be always the same, and we also keep the same first 24 bits of the cryptogram on 8-bytes for which the card answered.

3. We try all the 256 possibilities (255 remaining to be tried) for the 4th byte of the cryptogram. The second 4 bytes c_4, c_5, c_6, c_7 of the 8-byte cryptogram can be fixed or random, it does not matter. We get 256 cases, out of which in about $256/32 \approx 8$ cases the card will reply with a 4-bit encrypted NACK.

   Moreover, in the lucky event described in Fact 5.3.1, the number of those that will reply will be rather about $256/16 \approx 16$.

4. Now with probability about $1/12.8$, we can simultaneously predict the differences of the states for **all** the 256 cases (cf. Fact 5.4.1), where for some 8-16 cases we obtained 4 bits of ks3.

5. Now we will examine all the 2^{21} possibilities for the 21 bits of the state that determine bits $ks3_0$ and $ks3_2$. For each case where the card have answered we can divide the size of our state by 4. With about 16 answers expected, we will determine the value of the 21 bits in question with near-certainty.

   Or we will find a contradiction, which indicates that the keystream does depend on c_3, contrary to the assumption.

6. In the same way, we will determine another 21 bits of the state that determines bits $ks3_1$ and $ks3_3$.

7. If this fails, we change the 3th bit c_3, find another case where the card answers, which happens with probability $1/64$, and examine all the 256 cases for c_3 again. On average we have to repeat our attack with 256 queries 12.8 times.

5.6 Complexity Of Our Attack

The expected number of queries to the card in our attack is $256 + 11.8 \cdot 64 + 12.8 \cdot 255 \approx 4275$. This data takes about 30 minutes to gather if we assume the speed of two transactions per second.

The computation needed for this attack is of the order of 2^{21}. We have implemented the full attack, and once the data for the attack is available, it runs in a few seconds on a laptop.

This attack is so fast and requires so little memory, that it could be implemented in a small portable device that could be used to clone MiFare Classic cards in the card-only scenario.

5.7 Variants of Our Attack

This attack can have many variants, where with less queries to the card, and more computations, we can also recover the key. We can abort earlier on blocks of 256 queries that respond with lower frequency. We can run the attack for blocks of 64 consecutive c_3 values instead of 256, and the probability that we can predict all the 64 differences will be only 1/7.8. However then we expect only $64/16 = 4$ answers, and therefore we will only reduce the security of the cipher w.r.t. to a brute force attack by a factor of approximatively $2^{4.4} = 2^{16}$. This will result in an attack that runs in less than 1 hour on a laptop PC and requires about $256 + 64 + 6.8 \cdot (64 + 64) \approx 1200$ card queries, which requires to communicate with the card for about 10 minutes.

6 MORE ATTACKS

Other attacks have been announced by researchers (Garcia et al., 2009) and from NXP web page (NXP-web, 2008) we learn that:

"One particular card only attack can, with a certain prerequisite on knowledge about the card, retrieve all keys and data from the card in about a second per key using a laptop and limited value equipment. Interaction with the card can be limited to two times less than a second: first to get material for key recovery and then once the keys are retrieved an interaction to retrieve the data"

We don't have this paper and we cannot confirm if and how such attacks can be handled.

7 DEFENCES

Assuming that for most organisations it is infeasible (and would cost millions) to instantly replace their access control systems and their cards, the only defense against this attack is electromagnetic shielding: putting all the cards in a wallet that is shielded against electromagnetic fields. Such solutions have been developed for e-passports and are widely available. Ordinary aluminium foil can also be used.

Importantly, because we discovered that certain MiFare classic cards can be cloned in seconds, **we invite every institution or building that uses MiFare classic card, to check if their cards do not fall into this particular (much weaker) category.**

8 CONCLUSIONS

In our study and experimentation with MiFare Classic we (and other researchers) have discovered a security breach **far greater** than ever predicted. The property of millions of people, governments and businesses worldwide is at great risk. We have discovered that MiFare classic cards, as used to protect UK Cabinet Office building, London Underground, roughly probably about the half of the financial institutions in the City of London, and tens of thousands of buildings worldwide, can be cloned by anyone that is for example sitting next to the victim for some 2 minutes. This can happen at any moment, without raising the slightest suspicion, and can happen also when the victim travels to a foreign country.

We also developed a cryptographic shortcut attack: if we are allowed 10-30 minutes access to the card, which can still happen daily for example on an underground train, one can recover the keys inside MiFare Classic cards with a very small amount of computation.

Finally, we have found that for certain MiFare Classic 1 K cards, in particular certain cards found in Eastern Europe, and that otherwise seem **totally indistinguishable** in any respect from the popular MiFare Classic 1 K, the card can be cloned in a few seconds.

8.1 Lessons Learned

The classical model for smart card security is about hardware barriers that cannot be breached by software, physical control of the card, and trusting the entities involved in developing components of a secure system to enforce and defend their security perimeters correctly. **This model totally breaks apart** with RFID smart cards such as MiFare classic where the secrecy of the product is not an extra security layer, but a source of unexpected and critical security vulnerabilities that by the fact of being hidden, give an utterly false sense of security.

This vulnerability is a bug. **Or maybe** it is a backdoor intended to grant access to buildings with a criminal intention? We need to stress that the manufacturer of this system, NXP and formerly Philips, needs to be considered innocent unless proven guilty. In security research however, we can and should assume the worst possible scenario. The security industry clearly needs more supervision and accountability. More attention has to be paid into how security products are developed and how markets for security function. We need to invent new, or enhance the existing mechanisms (such as Common Criteria evaluations ISO 15408) for preventing this from ever happening again.

The main contribution of this paper is to show that trade secrets and the secrecy of cryptographic algorithms, protocols and security systems can be beneficial but also do have a dark side. More research needs to be done to understand all the large scale systemic threats to our economy that come from insider threats, electronic subversion, software and hardware bugs.

REFERENCES

de Koning Gans, G., Hoepman, J.-H., and Garcia, F. D. (2008). A Practical Attack on the MIFARE Classic. In *Procedings of the 8th Smart Card Research and Advanced Applications, CARDIS 2008*, LNCS.

Garcia, F. D., de Koning Gans, G., Muijrers, R., van Rossum, P., Verdult, R., and Wichers Schreur, R. (2008). Dismantling MIFARE Classic. In *Procedings of the 13th European Symposium on Research in Computer Security, ESORICS 2008*, LNCS.

Garcia, F. D., van Rossum, P., Verdult, R., and Wichers Schreur, R. (2009). Wirelessly Pickpocketing a Mifare Classic Card. In *Accepted at Oakland IEEE Symposium on Security and Privacy*.

Kerckhoffs, A. (1883). *La cryptographie militaire*, volume IX of *Journal des sciences militaires*.

NXP-web (2008). Security of mifare classic, official web page by nxp semiconductors.

Nohl, K. (2008). contains an open-source mifare classic implementation of mifare classic for ti trf7960 evm. personal web page.

Nohl, K., Evans, D., Starbug, and Plotz, H. (2008). Reverse-Engineering a Cryptographic RFID Tag. In *17th USENIX Security Symposium*, pages 185–194, San Jose, CA, USA. USENIX.

NXP-statement, P. (2008). on the court decision to allow the publication by radboud university nijmegen.

Rankl, W. and Effing, W. (2003). *Smart Card Handbook*. Wiley.

Rescorla, E. (2004). Is finding security holes a good idea? In *WEIS 2004, 3rd Workshop on the Economics of Information Security*.

Schneier, B. (2008). The ethics of vulnerability research. A blog covering security and security technology.

Schneier, B. and Shostack, A. (1999). Breaking up is hard to do: modeling security threats for smart cards. In *WOST'99: Proceedings of the USENIX Workshop on Smartcard Technology*, pages 19–19, Berkeley, CA, USA. USENIX Association.

Young, A. and Yung, M. (1996). The dark side of black box cryptography. In *Advances in Cryptology – CRYPTO'96*.