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, U.K.

Keywords: Access control, RFID, Contactless smart cards, MiFare Classic, London Oyster card, OV-Chipkaart, Industrial 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 about 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 access or 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) however our attack requires less queries to the card and does not require any precomputation. In addition, we discovered that certain versions or clones of MiFare Classic are even weaker, and can be cloned in 1 second.

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 governments, businesses and financial institutions worldwide.

1 INTRODUCTION

The MiFare Classic is the most popular contact-less smart card used to protect access to buildings worldwide and in public transportation. For more than 10 years the specification of these cards was kept secret. In 2008, two teams of researchers (Nohl et al., 2008; de Koning Gans et al., 2008) have more or less independently reverse engineered MiFare Classic and discovered several very serious attacks, in particular the Random Number Generators (RNG) both in the card and reader are flawed, and the product used an incredibly weak stream cipher Crypto-1 that can be broken in 0.1 s (de Koning Gans et al., 2008). As a result, one can clone the card if the attacker has access to the RF communications with a genuine cards reader (that knows the secret key). Can a system that is so badly compromised be shown to be even more insecure? The answer is yes, and we (and also other researchers (Garcia et al., 2009)) have discovered card-only attacks on MiFare Classic. In fact the card contains a very nasty implementation bug (sort of backdoor). This vulnerability alone allows one 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.

This paper is not only 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 poses a threat of a very 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 – potentially – a very major security breach. At the occasion we will revisit the questions of information assurance and secure smart card

T. Courtois N. (2009).
THE DARK SIDE OF SECURITY BY OBSCURITY - and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime.
In *Proceedings of the International Conference on Security and Cryptography*, pages 331-338
DOI: 10.5220/0002238003310338

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 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 on 4 bits, it may seem that there are 5 different Boolean functions. The same Boolean function is used several times and implemented in hardware in a different way. This is a curious implementation strategy, and certainly not the most cost-effective one. In addition, the LFSR taps that are used by the non-linear Boolean function are evenly spaced, using every second bit. By the combination of these two properties, the output of many Boolean functions inside the cipher are simply exactly equal to outputs of other (seemingly different but in fact identical) Boolean function, computed a few clocks 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. Indeed, the probability that out of five 4-bit Boolean functions, there are only two distinct Boolean functions is very small, it is of the order of $2^{-3 \cdot 2^4} = 2^{-48}$.

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.

3 DISCLOSE OR NOT?

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.

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 3 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. The secrecy of the algorithm is an important asset that really improves the security. For instance 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 and to disclose all their details. 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 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 alternatives. Thus 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. Finally, 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 other firms, 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 bad products and have mislead their customers about their security should suffer some sort of sanction.

4 MIFARE CLASSIC PROTOCOL

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 XX' or '61 XX' by which it starts the mutual symmetric-key authentication process between the card and the reader, with the key pertaining to the block number XX.
3. The card answers with a random n_T on 4 bytes,
4. The reader sends a cryptogram on 8 bytes which is $\{n_R\} = n_R \oplus ks1$ and $\{a_R\} = suc^2(n_T) \oplus ks2$.
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 XX.

Here n_R is the 32-bit nonce chosen by the reader, $\{n_R\}$ is the encryption of it, suc is a certain bijective function, and $(ks1, ks2, ks3)$ are the 96 bits of the keystream produced by the Crypto-1 stream cipher after being initialized with n_T and n_R. We refer to (de Koning Gans et al., 2008) for more details.

4.1 Nice Properties of this Protocol

We can observe that this protocol is such as to make card-only attacks very hard if not totally impossible, this including brute force attacks (!) 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 $(\{n_R\}, \{a_R\})$, 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 $(\{n_R\}, \{a_R\}) = (n_R \oplus ks1, suc^2(n_T) \oplus ks2)$ 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

Notation. Let $C = (c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ be the 8-byte cryptogram sent the card as in point 4, independently whether it comes from a genuine transaction where $C = (\{n_R\}, \{a_R\})$ or from a spoofed authentication attempt. Let $P_C = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$ be the 8 parity bits that are sent together with the 8 bytes of C.

Our key discovery is that:

Fact 4.2.1. *If we 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 over the choice of C, P_C 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, which are the 4 first bits of $ks3$.*

Moreover, we have were able to guess and confirm by computer simulations when exactly this event occurs. We have independently discovered what one can now also learn from (Garcia et al., 2009):

Fact 4.2.2. *The card answers with a 4-bit encrypted NACK if and only if the parity bits for the plaintext after decryption are correct.*

In this paper we ignore how exactly are these parity bits computed, and we only need to know that:

Fact 4.2.3. *Whatever is the spoofed cryptogram C, there is exactly one choice for the 8 parity bits P_C so the card will reply with a 4-bit encrypted NACK.*

4.3 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. We would like to stress that to the best our knowledge, these cards are indistinguishable from any other card we have seen. They do give the same ATR (Answer to Reset) equal to ”3B8F8001804F0CA000000306030001000000006A”, we have ATQA = ”0400” and sak = ”080000” and they do functionally behave EXACTLY the same.

It appears, that, following (Roel, 2009), these weaker cards are unlicensed clones of MiFare Classic. For example we have ordered some Fudan Microelectronics FM11RF08 cards from China and verified that these cards also reply with probability 1.

5 PRELIMINARY ANALYSIS

A pre-requisite for our attack is a strict control of the timing of the transaction. By doing so, due to a 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. In (de Koning Gans et al., 2008) we read that the accuracy of selecting a chosen n_T by controlling the timing with a Proxmark 3 device is 1 out of 10. In Section 2 of (Nohl et al., 2008) we read that with a custom version of OpenPCD firmware one can reproduce exactly the same n_T each time with near certainty. We achieve a similar result with a modified firmware for TI TRF7960EVM reader developed by Bingsheng Zhang at University College London, based on the code found in (Nohl, 2008).

5.1 On Small Variability of Keystream

We consider now the Crypto-1 cipher as it is used in decrypting the 8-byte cryptogram C. 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 a few bits in the 4th byte c_3. The answer is that it depends surprisingly little of $ks3$.

Fact 5.1.1. *The probability that the 3 bits of the keystream generated during the decryption of the last 3 bits of the 4th byte c_3 do NOT depend on these bits of c_3 is very high, about* 0.75.

Explanation: This probability is surprisingly high, as shown by our computer simulations. (even when the full 8 bits of c_3 are variable, this probability is still very high, about $1/17$). All 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.

5.2 Impact of Small Variability of ks1

Let us fix the card nonce n_T, and a 29-bit prefix of the 8-byte cryptogram C. Let us assume that the card answers as in Fact 4.2.1 for some spoof attempt with this 29-bit prefix and some P_C. This means that all the 8 parity checks are correct after decryption.

Now, let us look what happens when we vary the last (in order of decryption and in order of transmission over the air) 43bits of the c_3 in C. If we look at the 9 keystream bits involved in decryption of the byte 3 of the plaintext $\{n_R\}$ (transmitted as 9 bits), it is easy to see that the first 6 bits are always constant, and that the last 3 bits are variable. Nevertheless, by Fact 5.1.1, all these 9 bits will actually be constant with probability 0.75. Then the 9 bits of decrypted plaintext will be constant as well, simultaneously for 8 different encryptions with the same 29-bit prefix. To summarize:

Fact 5.2.1. *If we fix the card nonce n_T and a 29-bit prefix of C, with probability about* 0.75 *over C the 9 bits of the keystream generated in this process are constant simultaneously for 8 different encryptions. of C that share the same 29-bit prefix and vary the last 3 bits of c_3.*

5.3 A Multiple Differential Property

The next question is how to exploit this in a cryptographic attack. We look the specification of the Crypto-1 cipher as used in the MiFare Classic authentication protocol, see (Garcia et al., 2008), and we observe that:

1. Every bit of the LFSR in the future is equal to a fixed and known linear combination of past bits of the LFSR, and of the bits that are injected to the LFSR, that are bits of $UID \oplus TC$, that are all known to the attacker, and the bits of n_R, unknown to the attacker and potentially hard to predict.
2. However we have $n_R \oplus \{n_R\} = ks1$ and $\{n_R\}$ is known to (and even chosen by) the attacker. The difference is the keystream that remains hard to predict, but following Fact 5.2.1 the whole $ks1$ will be constant with probability 0.75 over the choice of the 29-bit prefix of C.

Thus from the differential point of view we get:

Fact 5.3.1. *If we assume that the Crypto-1 keystream generated during the decryption of the 4th byte c_3 is constant and does NOT depend on this byte, 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.

6 OUR CARD-ONLY ATTACK

In (Garcia et al., 2009) we find a card-only attack on MiFare Classic that requires 28500 queries to the card, and another one with 4000 queries but requires to pre-compute a very large table. In this paper we present a new attack that is clearly better than any of these: it does not require any pre-computation, is extremely fast, and requires a few hundred queries.

1. **Stage 1.** We send 128 queries (in expectancy) with a fixed or random n_T, and a random 8-byte cryptogram $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$, and a fixed or random P_C to get one case where the card answers with 4 bits.
2. **Stage 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 29 bits of the cryptogram on 8-bytes for which the card answered, we keep the second half of the cryptogram (c_4, c_5, c_6, c_7) fixed, so that only 3 bits are variable in C overall (8 cases), and also we keep fixed (the same) first 3 parity bits in P_C.
3. We try in order each of the 8 possible values for $\{n_R\}$ and different choices of the last 5 bits of P_C. For each ciphertext, following Fact 4.2.3 for exactly one of the 2^5 cases the card will reply, it replies with probability $1/32$ if we try at random, and replies after 16 steps on average (we move then to next valus for $\{n_R\}$). After $8 \cdot 16$ queries on average we get 8 answers with all the 8 possible consecutive values of $\{n_R\}$.
4. The eight 4-bit replies we get here gives us $8 \cdot 4 = 32$ bits of information about the 48-bit key.
5. Now with probability about 0.75, we can simultaneously predict the differences of the states for all the 8 encryptions (cf. Fact 5.2.1 and Fact 5.3.1).

6. Now we use the fact that the combined Boolean function of Crypto-1 reuses most state bits after 2 steps. Thus exactly (and only) 21 state bits determine the two keystream bits bits $(ks3)_0$ and $(ks3)_2$. We will examine all the 2^{21} cases and for each ciphertext where the card have answered we can divide the size of our space by 4. With 8 answers we will determine about 2^5 possible values for the 21 bits.
7. In the same way, we will determine 2^5 possibilities for the other 21 bits of the state that determines bits $(ks3)_1$ and $(ks3)_3$.
8. Then we have a list of 2^{10} states on 42 bits, which we need to extend to 2^{16} possible states on 48 bits.
9. Then by simple roll-back we get about 2^{16} possible initial keys, and checking all the $8 \cdot 8$ parity bits involved in the attack allows to know which key is correct with near certainty.

   Or if we find a contradiction at any stage, this means that the keystream does depend on c_3, contrary to the assumption in Fact 5.2.1.
10. If this fails, we repeat the whole attack. On average we expect that the attack will be repeated $1/0.75 \approx 1.33$ times.

6.1 Complexity of our Attack

In repeated execution we can in fact save on the complexity of Stage 1: if we only change c_3 next time we ar at stage 1, the card will answer with probability $1/64$, and 32 queries on average are needed. Thus the expected average number of queries to the card in our attack is $128 + (1.33 - 1) \cdot 32 + 1.33 \cdot 8 \cdot 16 \approx 300$.

This data takes about 5 minutes to gather with our current setup, at a speed of one transaction per second. However if could implement it with the hardware configuration such as used in (Garcia et al., 2009), it should take only about 10 seconds.

For certain weaker cards as described in Section 4.3, we only need to query the card about 12 times (more than 8, here there is no parity bits to reject wrong keys) and repeat about $1/0.75$ times, which gives about 16 queries on average. These cards can be cloned in about 1 second.

Once the key for one sector is found by our attack, we can apply the nested authentication attack from (Garcia et al., 2009) to recover keys for all other sectors with a few queries per sector and with negligible running time.

The computation needed for this attack is of the order of 2^{22}. We have implemented the full attack, and once the data for the attack is available, the key is found in a few seconds on a laptop. We have verified that our attack works on a number of building passes and on a number of rail/underground passes from different countries. For example it is without any doubt safe to report that the content of data block 1 in every Oyster card (used in London underground and buses) is always the same: 0x96 followed by ABCDEFGHIJKLM and 0x01 twice.

Our 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: recover the key through interaction, read the card content, and directly copy to a pirate card. We are the first to propose such an attack.

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 1 second, we invite every institution or building that uses MiFare classic card, to check if their cards do not fall into this particular (weaker) category.

8 CONCLUSIONS

In our study and experimentation with MiFare Classic we (and other researchers) have discovered that the security is far worse than ever expected. The property of millions of people, governments and businesses worldwide is at risk. We have discovered that MiFare classic cards, used in London Underground, and reportedly in many government buildings, financial institutions, and many other buildings worldwide, can be cloned through invisible wireless interaction, by anyone that is for example sitting or standing next to the victim for a number of seconds. This can happen at any moment, without raising any suspicion.

One can try to compare our results to (Garcia et al., 2009), another paper exploiting the same vulnerability. It appears that our cryptographic shortcut attack based on a simultaneous conditional differential property is stronger than any of the attacks in (Garcia et al., 2009) and it is now the best known card-only attack on MiFare Classic. For all these attacks, the running time can be very fast, almost instant from the practical point of view. However, our attack requires only 300 queries, way less than any of the attacks described in (Garcia et al., 2009), and it does not require any precomputation.

Moreover, we have discovered that for some cards found in Eastern Europe and in China, thought they seem totally indistinguishable in any respect from other MiFare Classic cards, one only needs about 16 queries to the card to get the data needed in our attack.

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. Trade secrets and the secrecy of cryptographic algorithms and protocols can be beneficial but also do have a dark side. We need to invent new, or enhance the existing mechanisms (such as Common Criteria evaluations ISO 15408) for preventing this from ever happening again. 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*.

Nohl, K. (2008). Contains an open-source mifare classic implementation of mifare classic for ti trf7960 evm. personal web page, www.cs.virginia.edu/k̃n5f/.

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. www.mifare.net/news/statement_on_court_deci sion.asp.

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*.

Roel (2009). Mifare classic clones. Web forum www.proxmark.org/forum/topic/169/mifare-classic-clones/.

Schneier, B. (2008). The ethics of vulnerability research. A blog covering security and security technology, www.schneier.com/blog/archives/2008/05/ the_ethics_of_v.html.

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*.