CRYPTOGRAPHIC ENGINEERING ASSIGNMENT II
Theoretical: Design Weaknesses in MIFARE Classic

Özgecan Payzin, s4159721
ozgecan.payzin@student.ru.nl

April 1, 2013

1 Introduction

The MIFARE Classic is one of the most widely used contactless smart cards in the world. Designed and sold by NXP Semiconductors, MIFARE Classic smart cards are used for a variety of sensitive purposes, such as access control (including military and governmental restricted areas) and payment for public transport. The Dutch OV-Chipkaart system uses a MIFARE Classic, as well as the London Oyster card, and the Smart Rider in Australia. The smart card system uses authentication and encryption protocols to secure communications between the card and the reader and ensuring that the card has not been tampered with.

However, since 2008, a variety of extremely serious cryptographic, protocol and design flaws have been discovered in the MIFARE Classic that allow any attacker to read, copy and modify any information on the card. Some of these weaknesses have led to attacks that are able to retrieve the secret keys, modify information on the card without knowledge of the secret key, and potentially clone the card wirelessly without the card leaving the victims possession.

This paper will examine and summarize the weaknesses found in the MIFARE Classic, taking [1], [2], [3], and [4] as a basis. We will conclude with some brief examples of attacks that have been proposed and constructed using these weaknesses.

2 Structure of the MIFARE Classic

The memory of the MIFARE Classic is divided into logical sectors and data blocks. Each data block is 16 bytes, and each sector has four of these data blocks[1]. Individual sectors have their own 48 bit secret keys, of which there are two: Key A and key B. These secret keys are stored in the last data block of each sector; this block is called the sector trailer.

In addition to the keys, the sector trailer also holds data about the access conditions of the sector. These include what operations are allowed in the sector and what key must be used to access the sector. The first 6 bytes of the trailer are used for key A, followed by 3 bytes of access conditions, one data byte U with no defined purpose, and the last 6 bytes for key B.

[1] In the MIFARE Classic 1k, this is true, however, the MIFARE Classic 4k differs in that the first 32 sectors have 4 data blocks each and the last 8 have 16 data blocks each.

The sector trailer itself is subject to special access conditions. Key A is never readable and attempts to read it will return a plaintext of all zeros. Key B may or may not be readable, as defined by the access conditions. If it is not readable, attempts to read it will return a plaintext of all zeros. This means that the sector is only used for data storage, as there are no keys to authenticate for it.

In order for a reader to have access to data in a sector, the reader must first authenticate specifically for that sector. After the anti-collision protocol, in which the card[2] sends its UID to the user and the reader selects the card, the authentication protocol, taking the form of a set of challenge-response messages, begins. Midway through the authentication process, the messages start being sent in an encrypted form, and from that point onwards all messages exchanged are encrypted. The encryption is performed in the form of a one-time pad whose keystream is generated by 48-bit linear feedback shift registers (LFSR), whose output is then put through a non-linear filter function to yield one bit of keystream. After authentication is completed, encrypted commands and responses may be sent between the tag and the reader. A detailed explanation of the weaknesses in the authentication and encryption protocols follows.

3 Authentication

The authentication protocol of the MIFARE Classic proceeds as follows:

1. The reader sends an authentication request for a specific sector and key to the reader in the clear.
2. The tag sends a 32-bit tag nonce n_T as a challenge in cleartext to the reader.
3. The reader responds with an encrypted nonce n_R and the encrypted response to the tag challenge a_R.
4. The tag responds with the encrypted response a_T.

If the challenges have been responded to correctly, messages may now be exchanged between the tag and reader. This procedure must be repeated for each different sector the reader wants to access; however, after the first authentication process, the authentication request and n_T are both encrypted.

There are a variety of weaknesses in the authentication protocol. One weakness is that if the tag does not respond with a_T in a timely manner after the reader sends its challenge, the reader will send a halt command and stop the protocol. However, this halt command is encrypted with the keystream, regardless of the fact that the tag has not authenticated and presumably cannot decrypt it. Since the byte code for the halt command is known, this means part of the keystream can be recovered. Alternately, some readers react as if the authentication protocol is completed and typically send an encrypted read command instead. Since the byte code for the read command is also known, partial keystream recovery is possible by guessing which block the read command is meant for.

A similar weakness arises due to the parity bits used in the protocol to detect transmission errors. If the parity bits are correct but the response wrong during step 3 in the authentication protocol above, the tag will respond with an encrypted transmission error code. Since this code is known, it is possible to recover some keystream, in addition to leaking information about the message. Certain unlicensed versions of the MIFARE Classic always yield this error if the response is wrong regardless of the parity bits.

[2]From this point onwards, the terms card and tag will be used interchangeably.

Another problem during the authentication stems from the fact that the pseudo-random number generator (PRNG) on the tag is weak. The PRNG on the tag uses a 16-bit LFSR to generate tag nonces, however, the PRNG is fully deterministic and only the time between the power-up of the tag and the start of the communication determines the nonce that will be generated. This allows an attacker to repeatedly be able to send a desired tag nonce. Due to the weakness of the PRNG, it is also theoretically possible for the same nonce to naturally reappear and be re-used in as little as 0.6 seconds. This has further implications regarding the encryption process due to how strongly n_T influences the keystream, discussed in Section IV.

In addition, the fact that the PRNG changes predictably results in nonces n_T in subsequent authentication attempts for different sectors having a calculable relationship to one another, affected by the number of state changes of the LFSR in the PRNG. Thus, regardless of whether a subsequent n_T is sent encrypted, it can be recovered with a very high probability, along with the keystream used to encrypt it. This makes deciphering subsequent sector authentication requests simple.

A further problem with the PRNG is that due to the usage of a 16-bit LFSR to create 32-bit tag nonces, the second half of the tag nonce always depends on the first half. This leaks information about the structure of a proper tag nonce and narrows the space from which nonces can be picked, thus, in the cases that an exhaustive search for a specific tag nonce is necessary, this task becomes significantly easier.

This property also allows an attacker to recognize the structure of a proper tag nonce. In practice, this piece of knowledge was used along with weaknesses in the encryption scheme to determine that the responses a_T and a_R directly depend on n_T; in fact, a_R is equal to the second successor[3] of n_T and a_T is equal to the third successor of n_T. This has the effect of making the authentication protocol very predictable and allows a portion of the initial keystream to be recovered, as the plaintexts are known.

The PRNG on the reader also has a weakness where it does not reset with every clock cycle but with every invocation, thus allowing an attacker to keep n_R the same every time the authentication protocol is started.

4 Section IV: Encryption

The MIFARE Classic uses Crypto1 as its cipher. Crypto1 is a proprietary algorithm that makes use of a 48-bit LFSR, a feedback function that also incorporates input from various other sources during initialization, and a non-linear filter function that reduces the state of the LFSR to a single bit.

Normally, at every clock cycle the leftmost bit is discarded and the rightmost bit is recalculated using the feedback function. However, during the initialization phase of the LFSR, the output from the feedback function is also XORed with a separate bit, termed the input bit. Keystream bits are created by putting specific LFSR bits through a filter function. This filter function is two-layered, in that the output of the first layer goes through a second layer function. The output of the second layer is a single bit of the keystream. The first bit of the keystream is required at the step of the authentication when the reader nonce n_R and the response a_R are sent back, as n_R needs to be encrypted at this point.

Crypto1 has been completely reverse-engineered during research into the security of the MIFARE Classic and has a number of severe weaknesses that nullify the security the cipher is meant to ensure. These weaknesses largely mean that given a specific keystream, it is possible to obtain the state of the LFSR at the time the keystream was created. It is then possible

[3]A successor is defined as the next 32 bits generated by the same LFSR that generated n_T.

to roll back the state of the LFSR until the initial state, that has been determined to be the secret key, is reached. Other weaknesses allow for discovery of the keystream without knowing the secret key, or the ability to control the internal state of the LFSR.

An initial observation regarding the Crypto1 cipher was that if the quantity $n_T \oplus$ UID was kept constant, the ciphertext containing the encrypted reader nonce would also stay constant. Since the tag nonce is trivial to keep constant due to the weaknesses discussed in Section III, this effect can reliably be reproduced using the same card. This leads to the logical conclusion that the LFSR incorporates $n_T \oplus$ UID as an input into the LFSR after its initial state.

Another observation was that if the combination XOR of n_T, UID, the secret key K and the feedback bits would be kept constant, then the keystream generated after authentication is completed also stays constant. This leads to two important conclusions: One, that the secret key K is almost certainly the initial state of the LFSR, and two, it is possible to control the state of the LFSR at the point right after $n_T \oplus$ UID has been fed in by keeping K, n_T and UID constant. All of these quantities can be controlled and kept the same by the attacker.

It may seem that the leftmost 16-bits of the 48-bit LFSR at this point cannot be completely controlled by the attacker, as the tag nonce is 32-bits, so the leftmost 16 bits will always be whatever the UID is set to. However, many readers have a weakness where they will accept tag nonces longer than 32 bits, so feeding in a 48 bit tag nonce allows for complete control.

The second set of problems involving the encryption scheme stem from the filter function of the LFSR. The filter function uses a limited amount of bits from each state of the LFSR to generate the keystream, so the output at any given state depends on only a small portion of the LFSR. In addition, only the odd numbered bits are used in the filter function, and they are evenly spaced. These weaknesses can be used to construct an attack to reverse the filter function, i.e go from a given keystream to the LFSR states that can create such a keystream.

Another important weakness that makes the rollback process possible is that the leftmost bit of the LFSR is never used in the filter function and as such has no effect on the keystream. This makes it possible to arbitrarily assign this bit when the LFSR is being rolled back, i.e shifted one bit to the right instead of the left. The bit can then successfully be calculated for the next step from the available keystream and feedback function information.

A final problem with the LFSR is that the reader nonce n_R is fed as an input to the LFSR at the same time that the keystream to encrypt n_R is created. As a result, the initial bits of n_R directly affect the encryption of the later bits. This, combined with the rollback vulnerability described above, can be used to recover the initial state of the LFSR (i.e the secret key) if the state right after using n_R as the input is known.

The final set of problems with the encryption process involve the parity bits used in transmission. In compliance with the ISO standard, for every 8 bits of transmission, a parity bit is sent. This bit is used for error checking. The parity bit is calculated over the plaintext, rather than the ciphertext, which is insecure and in violation of the standard.

The parity bit itself is also encrypted, however, instead of using a new bit of keystream, the parity bit is encrypted with the keystream bit that is also used for the bit next in sequence. This violates the fundamental rule of one-time pads, which is to never reuse keystream, and leaks information about one bit per byte. This leakage can be used to drastically narrow the search space for encrypted tag nonces in subsequent sector authentication attempts, as some bits of the plaintext become known. It should be noted that in practice this is not necessary to use, as it is simple to find the encrypted tag nonce due to the relationship between subsequent tag nonces mentioned in section 3.

A final weakness is caused by the known plaintext in sector trailers and block 0 of sector 0. The structure and contents of the sector trailer are known, as noted in Section II. The structure of block 0 of sector 0 is also completely known: This block includes the UID of the tag and

a checksum, which are both known, as well as manufacturer data. Some of the manufacturer data is also known or guessable. By capturing one genuine transaction between a reader and a tag, it is possible to recover keystream by reading block 0 and block 3, and using the known plaintext to decode different parts of the keystream. This allows the entirety of sector 0 to be read without knowing the key. This weakness can also be generalized to read other sectors provided that the data within one block of the sector is known, using a known plaintext attack to send commands where the response is predictable and recovering more keystream.

5 Section V: Attacks

In this section we will briefly cover some theoretical and practical attacks that have leveraged the weaknesses above. Note that this section is not exhaustive and is merely meant to show examples of possible attacks.

An attack proposed in [1] takes advantage of the fact that the tag nonce directly influences the keystream, and thus, the internal state of the LFSR. The attacker first builds a table consisting of (LFSR, KS) pairs where LFSR is the 48-bit state of a specific form 0x000WWWWWWWWW and KS is the keystream generated by this state. Then, a tag nonce is chosen in the form of 0x0000XXX0, and the authentication protocol is started with the same UID. However, after receiving n_R and a_R, the tag does not respond, causing an encrypted halt command to be sent. At this point, the attacker leverages the successor weakness on a_R to calculate a_R and recover part of the keystream, and use the fact that the plaintext of the halt command is known to recover another part.

After n_R is used as an input to the LFSR, the LFSR state is now of the format 0x000WWWW WWWWW due to the specifically chosen nonce, and all states of this format are in the table of the attacker with their corresponding keystreams. The discovered partial keystreams may then be used to determine the internal state of the LFSR. At this point an attacker may create the rest of the keystream and communicate with a device. It is also possible to obtain this state using the weaknesses that allow the filter function to be inverted, as the keystream at that point is known.

To extend on this attack, the rollback weakness may be used to discover the secret key K for that sector. The LFSR is shifted to the right, and the leftmost bit is assigned arbitrarily. Then the attacker can use the filter function to recover one bit of the keystream. This is the bit that is used to encrypt the last bit of n_R, so the last bit can be decoded. Using the input of this last bit and the feedback function, the leftmost bit from the previous step can now be calculated and the complete previous state obtained. As the previous inputs to the LFSR (the tag nonce n_T and the UID) are known in plaintext, the LFSR state can be rolled back in this manner all the way to its initial state, i.e the secret key.

In another attack using a reader controlled by the attacker and a genuine card, it is possible to read and modify card contents without having access to the secret key. This first requires the attacker to eavesdrop on a legitimate session. Afterwards, the attacker waits until the same tag nonce is eavesdropped in communication again, which will not take a long time. Since the card stays the same,the UID and n_T are now the same as the first session. As the reader is controlled by the attacker, the attacker can ensure that n_R is also the same, thus, the keystream will be exactly the same as the previous session. At this point the attacker can replay commands from the previous session, or modify plaintext and send commands to the card to which the answer is predictable, and use this known plaintext to recover keystream. Different parts of the keystream may be recovered by shifting the known plaintext.

The information revealed by the parity bits can be used in a brute force attack to reveal the key. The attacker, controlling a reader, makes authentication attempts for a sector, and

sends back a random n_R and a_R with 8 randomly chosen parity bits. This is repeated until an encrypted transmission error is received signaling that the parity bits are correct. At this point 12 bits of information regarding the keystream has leaked, 8 bits from the parity bits and 4 bits from the error message. Using this information a 48 bit key can be bruteforced, as approximately 6 successful attempts on parity bits will uniquely determine a key.

Another attack focuses on reader nonces n_R that have a specific property, resulting in a chosen ciphertext attack. The attacker must once again run an authentication session guessing the correct parity bits, then use this information to repeat the authentication varying the last encrypted bit of n_R. At this point the parity of the last byte has changed and the changed parity has been encrypted with the flipped version of the previous keystream bit. This information can be used to determine whether the last bit of n_R affects the encryption of the next bit. Only a limited number of the internal states of the LFSR have this property (approximately one tenth), and they can be brute-force searched to determine which state yields the correct keystream. This property can be generalized to each of the last bits of each byte of n_R, narrowing the search space for the internal state even more: if a nonce n_R is found whose each byte satisfies the property, there are approximately $7.3x10^9$ possibilities for the internal state.

References

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

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

[3] Garcia, F.D., van Rossum, P., Verdult, R., Schreur, R.W. (2009). Wirelessly Pickpocketing a Mifare Classic Card. In *IEEE Symposium on Security and Privacy*, pp. 315. IEEE, Los Alamitos.

[4] Courtois, N.T. (2009). The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime. In *SECRYPT 2009, International Conference on Security and Cryptography*, Milan, Italy, July 7-10.