Chameleon: A Versatile Emulator for Contactless Smartcards*

Timo Kasper, Ingo von Maurich, David Oswald, Christof Paar

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{timo.kasper, ingo.vonmaurich, david.oswald, christof.paar}@rub.de

Abstract. We develop a new, custom-built hardware for emulating contactless smartcards compliant to ISO 14443. The device is based on a modern low-cost microcontroller and can support basically all relevant (cryptographic) protocols used by contactless smartcards today, e.g., those based on AES or Triple-DES. As a proof of concept, we present a full emulation of Mifare Classic cards on the basis of our highly optimized implementation of the stream cipher Crypto1. The implementation enables the creation of exact clones of such cards, including the UID. We furthermore reverse-engineered the protocol of DESFire EV1 and realize the first emulation of DESFire and DESFire EV1 cards in the literature. We practically demonstrate the capabilities of our emulator by spoofing several real-world systems, e.g., creating a contactless payment card which allows an attacker to set the stored credit balance as desired and hence make an infinite amount of payments.

Keywords: RFID, contactless smartcards, payment systems, access control, efficient implementation

1 Introduction

Radio Frequency Identification (RFID) devices are deployed in a wide range of transportation and access control systems world-wide. If high privacy or security demands have to be met, typically contactless smartcards according to the ISO 14443 standard [13] are employed, as they offer sufficient computational power for cryptographic purposes. Moreover, a growing number of payment systems incorporates secure RFID cards [16], as they offer additional benefits in terms of flexibility and convenience over their contact-based counterpart. State-of-the-art contactless cards, such as the electronic passport ePass [8], provide a high level of security by means of various cryptographic primitives.

In general, RFID technology implies new threats compared to contact-based systems, for instance, a card residing in a pocket or wallet could be read out or modified without the owner taking note of it. Due to the cost sensitivity of such high-volume applications, card manufacturers are tempted to use outdated but "cheap" cryptographic components, e.g., in Mifare Classic products.

Since the reverse-engineering of the Crypto1 cipher used in Mifare Classic cards and the subsequently published attacks (cf. Sect. 3.1), the cards have to be regarded as insecure, as the secret keys can be extracted in seconds by means of card-only attacks. Once all keys of the card are known to an attacker, cards can be modified or duplicated. As many systems in the real world still rely on these weak cards, severe security threats may arise.

Accordingly, recently installed contactless systems, especially those with high security demands, are based on the DESFire variant of the Mifare family, and system integrators

* The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

upgrade the old Mifare Classic technology to these newer cards wherever possible. While the 3DES cipher employed in these cards is secure from the mathematical point of view, the implementation on the card is vulnerable to side-channel analysis, so that it is again possible to extract the secret keys of a card[1], as detailed in Sect. 3.2. Hence, emulating these modern cards is also practical and renders various attacks in real-world scenarios possible.

The resulting security weaknesses can become very costly – one example is a widespread contactless payment system based on Mifare Classic cards as analyzed in [16], where the credit value on the cards can be modified by an adversary with minimal efforts. For many of these systems, the read-only Unique Identifier (UID) of each card constitutes the only means to detect fraud in the backend, as there are no cards available on the market where the UID can be altered. In this paper, we exhibit the possibility of emulating and cloning RFID-enabled smartcards compliant to ISO 14443, including their UID.

1.1 Background and Related Work

Several research groups have proposed custom devices to emulate and counterfeit RFID devices. However, virtually all emulators presented so far suffer from certain drawbacks, e.g., insufficient computational resources, high cost, or impractical dimensions, limiting the threat they pose in the context of attacking real-world systems.

A custom RFID emulation hardware called Ghost is presented in [24]. The Ghost is able to emulate Mifare Ultralight cards which do not use any encryption. Emulating contactless cards employing secure cryptography seems to be impossible using this device due to computational limitations. The OpenPICC project [20] is mainly an RFID sniffing device. There was an approach to offer support for ISO 14443A, but the project seems to be discontinued. The Proxmark III [21] enables sniffing, reading and cloning of RFID tags. Since the device is based on a Field Programmable Gate Array (FPGA), it is also capable of emulating Mifare Classic cards, but at a comparably high cost of $399. The "HF Demo tag" [12] is based on an Atmel ATMega128 microcontroller which is not computationally powerful enough to perform encryptions with state-of-the-art ciphers in the time window given by the relevant protocols. An embedded system for analyzing the security of contactless smartcards was introduced in [14]. The attack hardware consists of a so-called Fake Tag and an RFID reader and can be used for, e.g., practical relay attacks. The device is based on a Atmel ATMega32 [1] processor with a constrained performance and is designed such that all important functionality is provided by the RFID reader. Hence, in addition to the lack of computational power, the Fake Tag cannot operate independently from the reader, which can be a major drawback for practical attacks. The authors also implemented an emulation of Mifare Classic, but similar to the HF Demo tag, the encryption runs too slow so that timing constraints of the protocol cannot be met. We used this work as a starting point for the development of our new standalone RFID emulator.

1.2 Contribution of this Paper

We built a freely programmable low-cost device that is capable of emulating various types of contactless smartcards, including those employing secure cryptography. The device operates autonomously without the need of a PC, can be powered from a battery, and possesses an Electronically Erasable Programmable Read-only Memory (EEPROM) for storing received bitstreams or other non-volatile information. An attacker using the presented hardware, which can be built for less than $25, is in full control over all data stored on the emulated card, including its UID and the secret keys.

[1] Note that the effort for extracting secret keys from Mifare DESfire cards by means of side-channel analysis is much higher compared to the Mifare Classic attacks.

In order to demonstrate the capabilities of our emulator in the context of real-world attacks, we implemented optimized versions of the Crypto1 stream cipher, the Data Encryption Standard (DES), Triple-DES (3DES) and the Advanced Encryption Standard (AES), as required for emulating the widespread Mifare Classic, Mifare DESFire and Mifare DESFire EV1 cards. With the developed software, it is possible to simulate the presence of one of these cards with an arbitrarily chosen content and identifier, and hence spoof real-world systems in various manners. For example, the emulator can behave as a card that automatically restores its credit value after a payment, or that possesses a new UID and card number on each payment, which impedes the detection of fraud. Besides the simulation of cards, our hardware allows for sniffing, e.g., reverse-engineering of protocols, relay attacks, and testing the vulnerability of RFID readers towards a behavior of the card that does not conform to the specifications, for instance, with respect to timing, intentionally wrong calculation of parity bits, or buffer overflows.

The remainder of this paper is structured as follows: in Sect. 2, we present our custom RFID hardware that serves as a basis for card emulations and attacks. After giving a brief summary of the relevant characteristics and protocols of Mifare Classic, Mifare DESFire and Mifare DESFire EV1 cards in Sect. 3, we detail on our implementations of the respective emulations in Sect. 4. Finally, practical real-world analyses performed with our hardware are described in Sect. 5.

2 Hardware Setup

In the following, we give a brief introduction to the physical characteristics of the RFID technology employed in contactless smartcards. Then, our freely programmable emulator for contactless smartcards is presented.

2.1 RFID Technology

In a typical setup for contactless smartcards, a reading device generates a strong Electro-Magnetic (EM) field at a frequency of 13.56 MHz for supplying the card with energy for its operation. The reader acts as master, while the card serves as slave, thus only the reader can start a communication and issue commands to the card. The ISO 14443 standard specifies the physical characteristics, the data modulation and other characteristics of contactless smartcards. For data transmission, the reader encodes the bits using a pulsed Miller code and transmits it by switching off the EM field for short periods of time. The data to be sent by the card is encoded using a Manchester-code and is afterwards transmitted via the EM field using load-modulation with a 847.5 kHz sub-carrier.

2.2 Our Emulator

For the security analyses in this paper, we developed a custom, freely programmable device termed “Chameleon”, which can emulate contactless smartcards compliant to the ISO 14443 standard in a stand-alone manner. Our emulation device consists of off-the-shelf hardware and can be built for less than $25. It is based on an Atmel ATxmega192A3 microcontroller [2, 3] which provides 192 kB of program memory, 16 kB SRAM and 4 kB EEPROM memory. Using an FTDI FT245RL chip [9], the ATxmega is able to communicate with a PC via the Universal Serial Bus (USB). This communication link can be used for debugging purposes and data manipulation at runtime. Figure 1 shows the first version of our RFID emulation device.

We chose the ATxmega because it features a hardware acceleration of both DES and AES-128. After loading the key and the data to the corresponding registers, the ATxmega is able

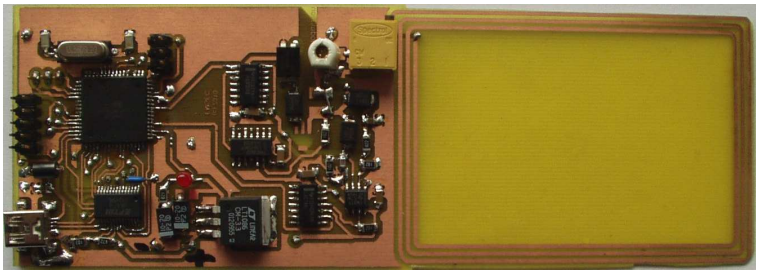

Fig. 1. Our stand-alone RFID emulation device.

to perform a DES en- or decryption in 16 clock cycles, i.e., one DES round per clock cycle, whereas the AES engine runs concurrently to the CPU and requires 375 clock cycles until an en- or decryption of one block is finished. The microcontroller is clocked by an external 13.56 MHz crystal, which is internally doubled using a high frequency Phase Locked Loop (PLL).

The coupling to the reader is established by a rectangular coil on the Printed Circuit Board (PCB). Variable capacitors are placed in parallel to form a parallel resonant circuit that is tuned to the carrier frequency. Analog circuitry assists the microcontroller in extracting the encoded data from the EM field and transmitting bitstreams. The design is similar to [14] and mainly shapes the signals according to the ISO 14443 standard and converts them to the appropriate voltage levels. Our emulation device can either be powered via the USB interface or run on battery. As all functionality is directly provided by the microcontroller, the Chameleon operates autonomously without the support of a PC. The full schematics of the developed hardware are given in the Appendix B.

3 Mifare Cards

This section covers the details of Mifare Classic, DESFire and DESFire EV1 cards. We present important facts required for the emulation of the cards and detail on the different authentication protocols, as implemented in Sect. 4.1 and Sect. 4.2. For reference, the complete protocols including the command codes and the low-level format are provided in Appendix A.

3.1 Mifare Classic

Since its introduction more than a decade ago, allegedly over 1 billion Mifare Classic ICs and 7 million reader components have been sold [18]. The cards provide data encryption and entity authentication based on the proprietary stream cipher Crypto1 for preventing from attacks like eavesdropping, cloning, replay and unauthorized reading or modification of the data stored on the card. Crypto1 is based on a Linear Feedback Shift Register (LFSR) with a length of 48 bit.

Basically, a Mifare Classic card can be regarded as a secured EEPROM memory with an RFID communication interface. In this work, we focus on the by far most widely employed Mifare Classic 1K version with 1024 byte EEPROM. All Mifare Classic variants comply to Parts 1-3 of ISO 14443A [13]. While the standard also allows for higher data rates, the cards communicate at a fixed data rate of 106 kBit/s. In addition, they feature a proprietary high-level protocol that diverges from Part 4 of ISO 14443A.

The memory of a Mifare Classic card is divided into sectors, whereas each sector consists of four blocks, as illustrated in Fig. 2. Each sector can be secured by means of two cryptographic

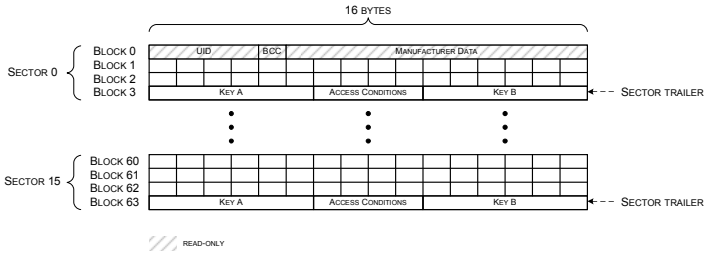

Fig. 2. The memory structure of a Mifare Classic 1K card.

keys A and B that are stored along with a set of access conditions in the last block of each sector. Before a sector can be accessed, a proprietary mutual authentication protocol with the appropriate secret key has to be carried out, cf. Protocol 1. The access conditions determine the commands that are allowed for each block of the sector (read, write, increment, decrement) and define the role of the keys [19]. The other blocks of each sector can be used for data storage. Note that the first block of the first sector differs from this scheme: it always contains a UID, along with some other manufacturer-specific data. The first block is written to the chip at manufacturing time, making it impossible to change the UID.

When a card is placed close to a reader, the anticollision and select procedure as defined in ISO 14443A is carried out. Then, an authentication command is issued by the reader that specifies for which sector the authentication is performed. The card replies with a 32-bit nonce n_C generated by its internal Pseudo-Random Number Generator (PRNG). The reader replies with an encrypted nonce n_R and an answer a_R, which is generated by loading n_C into the PRNG and clocking it 64 times. For the encryption, the keystream generated by the Crypto1 cipher is used in groups ks_1, ks_2, ... of 32 bit each. After the card has sent the encrypted answer a_C, both parties are mutually authenticated. From that point onwards, the reader can read, write or modify blocks in the chosen sector. If another sector has to be accessed, the authentication procedure must be repeated with a slightly modified protocol.

Reader		**Card**
	$\text{AUTH} \,\|\|\, \text{sector} \longrightarrow$	
		$n_C \in_R \{0,1\}^{32}$
	$\longleftarrow n_C$	
$n_R \in_R \{0,1\}^{32}$		
$a_R = PRNG(n_C, 64)$		
	$n_R \oplus ks_1 \,\|\|\, a_R \oplus ks_2 \longrightarrow$	
		$a_R \stackrel{?}{=} PRNG(n_C, 64)$
	$\longleftarrow a_C \oplus ks_3$	
$a_C \stackrel{?}{=} PRNG(n_C, 96)$		

Protocol 1. The Mifare Classic authentication protocol.

Security of Mifare Classic Since its invention, the internal structure of Crypto1 was kept secret and no open review process was performed. The cipher and its PRNG were later recovered by [17] using low-cost hardware reverse-engineering techniques. The authors

pointed out several design flaws, i.e., the short key length of 48 bit, mathematical weaknesses in the feedback functions of the LFSR, the weak 16-bit PRNG and the fact that the nonce generated by the PRNG depends on the time elapsed between power-up of the card and the authentication command. Subsequently, strong attacks on Mifare Classic were published: an attack described in [7] utilizes a fixed timing to generate the same nonces for repeated authentications and obtain parts of the keystream. A method to recover a secret sector key is proposed in [10], requiring two recorded genuine authentications to one sector. The most powerful attacks are card-only attacks as presented in [11] and [5]. They exploit amongst others the weakness that a card sends an encrypted NACK (`0x5`) each time the parity bits of the message $n_R \oplus ks_1 \,||\, a_R \oplus ks_2$ are correct but the decrypted a_R is not (cf. Protocol 1). This reveals four bits of keystream with a probability of $\frac{1}{256}$. Finally, a secret key of a Mifare Classic smartcard can be extracted within seconds using a combination of card-only attacks as proposed in [16], hence the cards can be considered fully broken.

3.2 Mifare DESFire and DESFire EV1

Mifare DESFire and Mifare DESFire EV1 cards are compliant to Parts 1-4 of ISO 14443A. Their UID is seven bytes long, and they support high baud rates of up to 848 kBit/s. A communication with the cards can be performed in plain, with an appended Message Authentication Code (MAC), or with full data encryption. Mifare DESFire cards offer 4 kByte of storage and data encryption by hardware DES and 3DES encryption. Mifare DESFire EV1 cards additionally provide AES-128 data encryption and are sold in three variants with 2 kByte, 4 kByte and 8 kByte of non-volatile memory, respectively. Each card holds up to 28 different applications with up to 14 different keys per application. For DESFire, each application may contain up to 16 files, while for DESFire EV1 the maximum number of files is 32. As in Mifare Classic cards, the UID is unchangeably programmed into the card at production time. Depending on the access rights for each application a mutual authentication protocol (see Protocol 2 / Protocol 3), ensuring that the symmetric key of the card K_C and of the reader K_R are identical, has to be completed before reading and manipulation of the data.

Reader		**Card**
	$\xrightarrow{\text{AUTH (02 0A 00)}}$	$n_C \in_R \{0,1\}^{64}$
$n_R \in_R \{0,1\}^{64}$	$\xleftarrow{b_0}$	$b_0 = Enc_{K_C}(n_C)$
$b_1 = Dec_{K_R}(n_R)$		
$r_0 = Dec_{K_R}(b_0)$		
$r_1 = RotLeft_8(r_0)$		
$b_2 = Dec_{K_R}(r_1 \oplus b_1)$	$\xrightarrow{b_1, b_2}$	$r_2 = Enc_{K_C}(b_2)$
		$n'_C = RotRight_8(r_2 \oplus b_1)$
	$\xleftarrow{\text{ERROR (02 AE)}}$	if $n'_C \neq n_C$
		else if $n'_C = n_C$
		$r_3 = Enc_{K_C}(b_1)$
		$r_4 = RotRight_8(r_3)$
	$\xleftarrow{b_3}$	$b_3 = Enc_{K_C}(r_4)$
$r_5 = Dec_{K_R}(b_3)$		
$n'_R = RotLeft_8(r_5)$		
verify $n'_R \stackrel{?}{=} n_R$		

Protocol 2. The Mifare DESFire authentication protocol [4].

Previous to the authentication, an application represented by its Application Identifier (AID) is selected. The reader starts the authentication protocol [4] with an authenticate command together with the key number that is to be used during the authentication. Note that Mifare DESFire cards only perform (3)DES *en*cryptions $Enc_K(\cdot)$ employing the secret key K, hence, DESFire readers always have to use (3)DES *de*cryption $Dec_K(\cdot)$.

As illustrated in Protocol 2, a DESFire card responds to the authentication command with an encrypted 64-bit random nonce n_C. The reader likewise chooses a 64-bit random nonce n_R, decrypts the received n_C, rotates it eight bits to the left and decrypts n_R as well as the rotated n_C. The card verifies if the rotated value equals n_C after reverting the rotation. If so, the card encrypts the first value to obtain n_R, rotates it eight bits to the right and encrypts the result which is then sent to the reader. The rotated and encrypted nonce is verified by the reader and if this final step is successful, both parties are mutually authenticated.

We furthermore reverse-engineered the DESFire EV1 authentication protocol, as presented in Protocol 3, by eavesdropping on genuine protocol runs. We found that the protocol of Mifare DESFire EV1 cards using AES-128 diverges from Protocol 2 as follows. In Protocol 3, en- and decryption are used in the common sense, i.e., data that is to be sent is encrypted and data that was received has to be decrypted. The CBC mode is modified in a way that all en- or decryptions are chained, even though they operate on different cryptograms. The Initialization Vector (IV) is not reset when en- or decrypting a new message, but instead depends on the previous en- or decryption. The nonces are extended to a length of 128 bit to match the block size of AES-128 and the second rotation is executed in the opposite direction on both sides. Again, AES-128 en- and decryption involving the key K are denoted by $Enc_K(\cdot)$ and $Dec_K(\cdot)$, respectively. Apart from that, the protocol equals the authentication protocol of Mifare DESFire cards and thus mutually authenticates both parties on successful execution.

Reader		**Card**
	$\xrightarrow{\text{AUTH (02 0A 00)}}$	$n_C \in_R \{0,1\}^{128}$
$r_0 = Dec_{K_R}(b_0)$	$\xleftarrow{b_0}$	$b_0 = Enc_{K_C}(n_C)$
$r_1 = RotLeft_8(r_0)$		
$n_R \in_R \{0,1\}^{128}$		
$b_1 = Enc_{K_R}(n_R \oplus b_0)$		
$b_2 = Enc_{K_R}(r_1 \oplus b_1)$	$\xrightarrow{b_1, b_2}$	$r_2 = Dec_{K_C}(b_1)$
		$r_3 = Dec_{K_C}(b_2)$
		$n'_C = RotRight_8(r_3 \oplus b_1)$
	$\xleftarrow{\text{ERROR (02 AE)}}$	if $n'_C \neq n_C$
		else if $n'_C = n_C$
		$r_4 = RotLeft_8(r_2 \oplus b_0)$
$r_5 = Dec_{K_R}(b_3)$	$\xleftarrow{b_3}$	$b_3 = Enc_{K_C}(r_4 \oplus b_2)$
$n'_R = RotRight_8(r_5 \oplus b_2)$		
verify $n'_R \stackrel{?}{=} n_R$		

Protocol 3. The Mifare DESFire EV1 authentication protocol.

Security of Mifare DESFire / EV1 The non-invasive side-channel attacks on RFID devices presented in [15] allow to extract secret information from contactless cards by measuring the electromagnetic emanations of a card while it carries out a cryptographic operation. The

focus is on devices that make use of DES or 3DES and the first successful key-recovery attack on such devices is accomplished. In a discussion with the authors, we came to know that the attacks have been improved since and are applicable to Mifare DESFire cards. With about 1 000 000 measurements they are able to fully recover the 3DES key stored on a Mifare DESFire card. Note that their side-channel attack is currently *not* applicable to DESFire EV1, which has been certified according to Common Criteria EAL 4+. However, efficient attacks might come up in the future or the secret key could obtained by other means, e.g., by exploiting weaknesses of the backend system.

4 Software Implementations

In this section we detail on our software implementations for emulating several cards with Chameleon.

4.1 Mifare Classic Emulator

The attacks detailed in Sect. 3.1 imply that an adversary can easily read out the secret keys and all content of a Mifare Classic card. To produce a duplicate, the adversary can write all previously read data to a blank Mifare Classic card. This results in an almost perfect clone, differing from the original only in the single block containing the read-only UID of the blank card. If the UID is verified by a contactless system (compare with [16]), this type of card-cloning becomes useless in practice. To allow for perfect clones, we implemented the features of Mifare Classic on Chameleon. Thus, we have complete control of the content of every memory block, including the previously unchangeable manufacturer block.

Optimized Crypto1 A first approach to emulate a Mifare Classic card on an AVR ATmega32 microcontroller [22] revealed difficulties in complying to the timing requirements given in ISO 14443. After a command is issued by the reader, the card has to reply within 4.8 ms, or the reader will reach a timeout and abort the connection. Compiling the open-source Crypto1 C-library [6] for an 8-bit microcontroller results in inefficient code regarding the underlying platform. Hence, in [22] the time limit of 4.8 ms set in the protocol is exceeded with 11.7 ms for an 18-byte encryption, neglecting all other necessary computations, such as encoding the encrypted data. Since an 18-byte encryption is required every time when reading or writing a block with appended CRC checksum, the existing implementation is not suitable.

It became obvious that a significant speedup of Crypto1 is essential for a successful Mifare Classic emulation. Hence, we implemented the cipher from the scratch in AVR assembly. This allows to optimize the code for an 8-bit platform and make use of special commands that may not be considered by the C compiler. Using instructions to access bits of registers directly, the amount of clock cycles required for an encryption was reduced, amongst others by replacing inefficient shifting and masking operations to access single bits with instructions that allow accessing a particular bit in one clock cycle (e.g., SBRC, BST, BLD). We further implemented the non-linear filter functions f_a, f_b and f_c of Crypto1

with lookup tables to avoid time consuming boolean AND, OR and XOR operations. In the first stage, f_a is used two and f_b three times with a 4-bit input of the state LFSR. Their output is used to generate a 5-bit input to f_c, which in turn generates one bit of keystream. For both f_a and f_b, we created a dedicated lookup table that includes the respective shifting of the output. Thereby, the input of f_c can be easily obtained by ORing the five outputs of f_a and f_b. This speed advantage comes at the cost of storing one bit of information in one byte of memory. Finally, the lookup table f_c is a simple 5-bit input, 1-bit output table. The overall size of the lookup tables is 112 byte, formed by two 16-byte tables for f_a, three 16-byte

tables for f_b and one 32-byte table for f_c. With respect to the 192 kByte size of the program memory, the tables are negligibly small.

Furthermore, we applied the idea of precomputation. When the nonce n_C is fixed before the authentication protocol is executed, the card is able to precompute the corresponding answers a_R and a_C which saves time during the authentication process. Precomputation of keystream bits is not possible because of two reasons. Firstly, since the sector to be accessed by the reader cannot be predicted, it is not clear which key has to be loaded into the LFSR. Secondly, the random reader nonce n_R that only becomes known during the authentication process is an input to the cipher.

4.2 Mifare DESFire (EV1) Emulator

Similarly to the Mifare Classic implementation, we additionally implemented the authentication protocols of both Mifare DESFire and Mifare DESFire EV1, as given in Sect. 3.2. For encryption, Mifare DESFire cards use DES/3DES in CBC mode, whereas Mifare DESFire EV1 cards can use either DES/3DES or AES-128 in CBC-mode.

4.3 Practical Results

Before carrying out security analyses in the real-world, we thoroughly tested our emulators in our laboratory. The reliability and accurate timing behaviour of our emulator was successfully verified with different RFID readers, including an ACG passport reader and a Touchatag [23] reader. Further tests with real-world systems are described in Sect.5.

Mifare Classic With the optimized implementation of Crypto1 detailed in Sect. 4.1, we successfully emulated Mifare Classic 1K cards with varying content. Table 4.3 summarizes the execution times for the relevant operations which are now all well within the limits specified in ISO 14443. All features, e.g., authentication, encrypted read and write of blocks, or specifying an arbitrary UID, are fully functional with the used readers.

Command	Execution time	Explanation
setup_crypto1()	98 μs	Initializes the cipher
auth_crypto1()	542 μs	Keystream for the authentication
crypto1_1()	8.3 μs	Generates 1 bit of keystream
crypto1_8()	49 μs	Generates 8 bits of keystream
crypto1_32()	186 μs	Generates 32 bits of keystream

Table 1. Execution times of crucial Crypto1 functions.

Mifare DESFire (EV1) Likewise, we tested our DESFire (EV1) emulations from Sect. 4.2. Table 4.3 shows the execution times for the needed cryptographic functions using the hardware accelerators of the ATxmega. Note that the first call to an en-/decryption function involves some overhead for the initial setup. After that, subsequent blocks can be processed faster. For reference, we included the runtime both for a single block and for ten data blocks in Table 4.3.

According to [4], an original Mifare DESFire card answers 690 μs (9356 clock cycles at 13.56 MHz) after b_1,b_2 was received when Protocol 2 is executed. During this time, two 3DES

encryptions are performed (one encryption of two blocks and one encryption of a single block). Our implementation performs about three times faster than a genuine card, with 219 μs (5932 clock cycles at 27.12 MHz) to produce a valid answer b_3 after b_1,b_2 was received.

Command	Block count	Execution time
TripleDES_CBC_Enc()	1 block	14.1 μs
TripleDES_CBC_Enc()	10 blocks	85.1 μs
AES128_CBC_Enc()	1 block	35.9 μs
AES128_CBC_Enc()	10 blocks	270.2 μs
AES128_CBC_Dec()	1 block	58.4 μs
AES128_CBC_Dec()	10 blocks	304.9 μs

Table 2. Execution times of 3DES and AES-128 en-/decryption functions.

A genuine DESFire EV1 card replies with b_3 approx. 2.2 ms after having received b_1,b_2. In contrast, our implementation only consumes about 438 μs and is thus faster by a factor of five. As we are able to en-/decrypt faster than both DESFire cards, encrypting or MACing data which is the most critical part for Mifare Classic does not pose a problem in the context of emulating DESFire (EV1) cards. For both Mifare DESFire and Mifare DESFire EV1, our implementation performed successfully with the readers in our laboratory. As with the emulation of Mifare Classic cards, we are able to equip our emulator with a UID that is free of choice.

We conclude that the ATxmega microcontroller on our current hardware revision is powerful enough to handle the amount of computation that is needed for the emulation of the simple Mifare Classic cards and also for more sophisticated contactless smartcards using 3DES or AES.

5 Real-World Attacks

We successfully employed the Chameleon to bypass the security mechanisms of several real-world systems, for example, we utilized the Mifare Classic emulation to fake a card that is accepted by a widespread payment system. In the following, we summarize the characteristics of this system and then detail on the attacks carried out with our hardware.

5.1 A Vulnerable Contactless Payment System

For the identification of a customer of the payment system analyzed in [16], in addition to the UID each card contains a card number chosen by the system integrator. The credit balance is stored in plain in a value block on the card, without any extra security measures. The credit can be increased by means of cash or a credit card at charging terminals, while the cash registers are equipped with RFID readers to decrease the credit according to the balance due. The contactless cards furthermore allow to open doors and grant access to restricted areas.

The system can be easily spoofed, because all cards issued have identical secret keys. Hence, once the secret keys of one card have been recovered, the content of any card in the system can be read out or modified. The authors were able to carry out payments by copying the content of original payment cards to blank Mifare Classic cards. The so obtained cards are not exact clones, since the UIDs of the blank cards are different from that of the genuine

ones, as detailed in Sect. 4.1. Consequently, the fraud could be easily detected in the back-end by verifying the correctness of the UID of a card on each payment.

The authors of [16] mention that the existence of a device that can fully clone a card including the UID would allow for devastating attacks, but suppose that these devices, if available, will be very costly so buying and using them for micropayments would not be profitable. With our developed hardware, the presence of an arbitrary valid card, e.g., an exact clone including the UID, can be simulated with minimal effort and cost, as shown in the following.

5.2 Electronically Spoofing a Contactless Payment System

A powerful type of attack that can be conducted with the Chameleon is called state-restoration. Even if the credit value was stored encryptedly on the payment card, e.g., using AES with an individual key per card, the content can be simply reset to the original credit value by dumping the full content of the card before paying and reprogramming the card (respectively our card emulation device) with the previous content after the payment.

As a first step to conduct this attack, we extracted the secret keys using the methods described in Sect. 3.1. Then, we dumped the content of a genuine card, including the UID, and copied it to our emulation device, thereby creating an exact clone. Hiding the device in a wallet, we consequently were able to carry out contactless payments. The credit value was stored in the EEPROM of our emulator and is decreased according to the balance due. As a result, the remaining credit displayed to the cashier appears to be correct and our device was accepted as genuine. The Chameleon allows to recharge the balance to its original value by restoring the initial dump when the attacker presses a push button. Finally, unlimited payments could be carried out with our device. Our practical tests furthermore showed that the Chameleon allows to open doors when cloning a valid card of an employee. However, if the fraud occuring due to the state restoration attack would be detected on the long term, the card number and/or the UID could be blacklisted and blocked for future payments.

For a more powerful attack, we programmed the Chameleon to generate a new random UID and card number for each payment. In our practical tests with the payment system, our emulator now appeared like a new card every time. Again, we were able to carry out payments, but this time, the device cannot be blacklisted and blocked in the backend.

In a similar manner, we were able to spoof a copy-and-print service that relies on contactless smartcards. The printers and copy stations are equipped with RFID readers that decrease the credit stored on the Mifare Classic card according to the amount of copies or printings carried out. By repeatedly using the service and comparing the content of the card between the payments, we found the block in which the amount of remaining credit was stored, again without any encryption. We hence programmed our card emulator to simulate the original card such that the credit appears to be lowered on each payment. However, the previous state of the card, i.e., charged to a high credit value, can again be restored by pressing a button on our hardware. As a consequence, we gain an unlimited amount of copies with our hardware.

Since cards of other customers can be read out from a distance[2], the Chameleon can also be used to clone their cards in a real-world scenario. Reading out the relevant sectors takes less than 100 ms. Several cards of other customers can be stored in the Chameleon and hence payments can be carried out with cloned cards that already exist in the payment system. Note that the original card of the customer remains unmodified and thus still contains the original credit value. Accordingly, a financial damage will only occur for the payment institution, while the customer is not affected. Altogether, taking the above illustrated devastating attacks and its low cost into account, the Chameleon can clearly be profitable for a criminal.

[2] Modified RFID readers allow for reading distances up to 30 cm

6 Conclusion

We present a microcontroller-based, freely programmable emulator for ISO 14443 compliant RFIDs that allows to simulate various contactless smartcards at a very low cost. The device works autonomously, operated from a battery, and its card-sized antenna fits into slots of most readers for contactless smartcards. Due to its small dimensions, the emulator can be used covertly, e.g., hidden in the purse, and is well-suited for real-world attacks. Our hardware can be connected to a PC by means of a USB interface and the non-volatile memory of the microcontroller allows amongst others to monitor the communication with an RFID reader and store the acquired data in order to reverse-engineer unknown protocols.

We exposed the protocol of Mifare DESFire EV1 cards, implemented the (3)DES and AES block ciphers as required, and present the first successful emulation of Mifare DESFire and DESFire EV1 cards in the literature. The current software further includes the emulation of Mifare Classic cards, based on a highly optimized variant of the Crypto1 stream cipher. The firmware of our device is not limited to Mifare cards but can be adapted to support other contactless smartcards and their respective protocols, e.g., the electronic passport and cards from other manufacturers.

We tested the emulations with different RFID readers and show that our implementations of the ciphers and protocols meet the timing requirements of all protocols and that the performance in most cases is even faster than that of original cards. In all our tests, the emulator could not be distinguished from a genuine card. The device proved to be a valuable tool for the security analysis of contactless technology and can be used to practically identify security weaknesses of real-world RFID systems.

Since secret keys of Mifare Classic cards and Mifare DESFire cards can be extracted by means of mathematical cryptanalysis and side-channel analysis, respectively, our emulator poses a severe threat for many commercial applications, if it was used by a criminal. To demonstrate the capabilities of our findings we perform several real-world attacks, amongst others on a contactless payment system. We emulate exact clones (including the UID) of Mifare cards, successfully spoofed an access control system and carried out payments. Furthermore, we implemented a mode of operation in which our emulator appears as a new card with a new UID and new content on every payment, which hinders detection of fraud in the backend.

With contactless payment, ticketing and access control systems being omnipresent today, it is crucial to realize that only strong cryptography, together with sound protocol design and protection against implementation attacks can ensure long-term security. Bug-fixes for broken systems based on false assumptions on certain device characteristics, e.g., UID-based protection schemes for Mifare Classic, are a fatal design choice, as we demonstrate that exact cloning of cards is feasible at a very low cost.

References

1. Atmel. ATmega32 Data Sheet. `http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf`.
2. Atmel. ATxmega192A3 Data Sheet. `http://www.atmel.com/dyn/resources/prod_documents/doc8068.pdf`.
3. Atmel. AVR XMEGA A Manual. `http://www.atmel.com/dyn/resources/prod_documents/doc8077.pdf`.
4. D. Carluccio. Electromagnetic Side Channel Analysis for Embedded Crypto Devices. Diplomarbeit, Ruhr-University Bochum, March 2005.
5. N. Courtois. The Dark Side of Security by Obscurity and Cloning Mifare Classic Rail and Building Passes, Anywhere, Anytime. In *SECRYPT 2009*, pages 331–338. INSTICC Press.
6. Crapto1. Open Implementation of Crypto1. `http://code.google.com/p/crapto1`, 2008.

7. G. de Koning Gans, J. Hoepman, and F. Garcia. A Practical Attack on the MIFARE Classic. In *Smart Card Research and Advanced Applications 2008*, volume 5189 of *LNCS*, pages 267–282. Springer.
8. Federal Office for Information Security, Germany. Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control. `http://www.bsi.de/fachthem/epass/EACTR03110_v110.pdf`.
9. Future Technology Devices International Ltd.. FT245R Datasheet. `http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT245R.pdf`.
10. F. Garcia, G. de Koning Gans, R. Muijrers, P. Van Rossum, R. Verdult, R. Schreur, and B. Jacobs. Dismantling Mifare Classic. In *ESORICS 2008*, volume 5283 of *LNCS*, pages 97–114. Springer.
11. F. Garcia, P. van Rossum, R. Verdult, and R. Schreur. Wirelessly Pickpocketing a Mifare Classic Card. In *Symposium on Security and Privacy*, pages 3–15. IEEE, 2009.
12. IAIK Graz. HF Demo Tag. `http://www.iaik.tugraz.at/content/research/rfid/tag_emulators`.
13. ISO/IEC 14443-A. Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part 1-4. `www.iso.ch`, 2001.
14. T. Kasper, D. Carluccio, and C. Paar. An Embedded System for Practical Security Analysis of Contactless Smartcards. In *WISTP 2007*, volume 4462 of *LNCS*, pages 150–160. Springer.
15. T. Kasper, D. Oswald, and C. Paar. EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In *WISA 2009*, volume 5932 of *LNCS*, pages 79–93. Springer.
16. T. Kasper, M. Silbermann, and C. Paar. All You Can Eat or Breaking a Real-World Contactless Payment System. In *Financial Cryptography 2010*, volume 6052 of *LNCS*, pages 343–350. Springer.
17. K. Nohl and D. Evans. Reverse-engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, pages 185–193, 2008.
18. NXP. About MIFARE. `http://mifare.net/about/`, 2001.
19. NXP. Mifare Classic 1K MF1 IC S50 Functional Specification. `www.nxp.com`, 2008.
20. OpenPICC. Programmable RFID-tag. `http://www.openpcd.org/openpicc.0.html`.
21. Proxmark III. A Radio Frequency IDentification Tool. `http://www.proxmark.org/`.
22. M. Silbermann. Security Analysis of Contactless Payment Systems in Practice. Diplomarbeit, Ruhr-University Bochum, November 2009.
23. Touchatag. Touchatag RFID Reader. `http://www.touchatag.com/`.
24. R. Verdult. Proof of Concept, Cloning the OV-Chip Card. `http://www.sos.cs.ru.nl/applications/rfid/2008-concept.pdf`.

A Authentication Protocols

This appendix provides the commands and the exact binary format for the authentication protocols used in this paper. Note that for DESFire (EV1), the message format according to ISO 14443A part 4 (including the 16-bit CRC) is taken into account in the following.

A.1 Mifare Classic Authentication Protocol

#	Direction	Protocol Message	Explanation
1	R → C	`60, sector (1 byte), CRC1 CRC2 (2 byte)`	Auth || sector || CRC
2	C → R	`4 byte`	n_C
3	R → C	`4 byte, 4 byte`	$n_R \oplus \mathrm{ks}_1 \,\|\|\, a_R \oplus \mathrm{ks}_2$
4	C → R	`4 byte`	$a_C \oplus \mathrm{ks}_3$

Table 3. Authentication protocol between a reader R and a Mifare Classic card C.

A.2 Mifare DESFire Authentication Protocol

#	Direction	Protocol Message	Explanation
1	R → C	`02 0A, key (1 byte), CRC1 CRC2`	Auth || key number || CRC
2	C → R	`02 AF, 8 byte, CRC1 CRC2`	Card nonce || b_0 || CRC
3	R → C	`03 AF, 8 byte, 8 byte, CRC1 CRC2`	Reader response || b_1 || b_2 || CRC
4	C → R	`03 00, 8 byte, CRC1 CRC2`	Success || b_3 || CRC

Table 4. Authentication protocol between a reader R and a Mifare DESFire card C.

A.3 Mifare DESFire EV1 Authentication Protocol

#	Direction	Protocol Message	Explanation
1	R → C	`02 AA, key (1 byte), CRC1 CRC2`	Auth || key number || CRC
2	C → R	`02 AF, 16 byte, CRC1 CRC2`	Card nonce || b_0 || CRC
3	R → C	`03 AF, 16 byte, 16 byte, CRC1 CRC2`	Reader response || b_1 || b_2 || CRC
4	C → R	`03 00, 16 byte, CRC1 CRC2`	Success || b_3 || CRC

Table 5. Authentication protocol between a reader R and a Mifare DESFire EV1 card C.

B Schematics

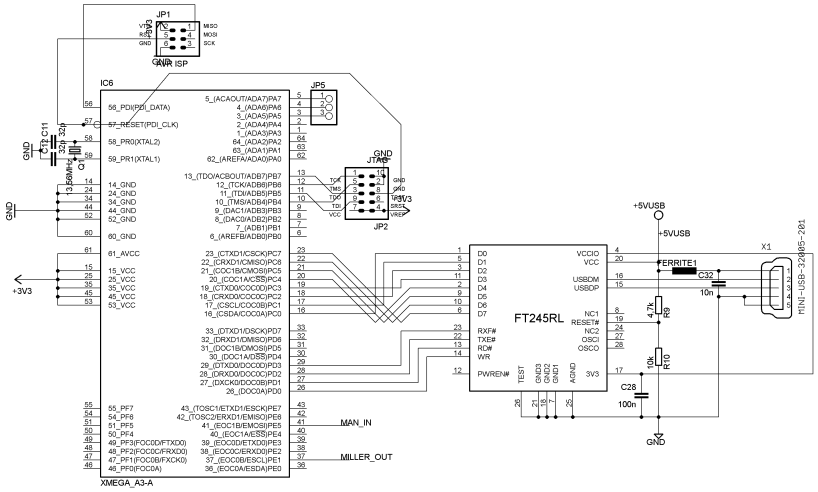

Fig. 3. Schematics of the microcontroller and the USB interface.

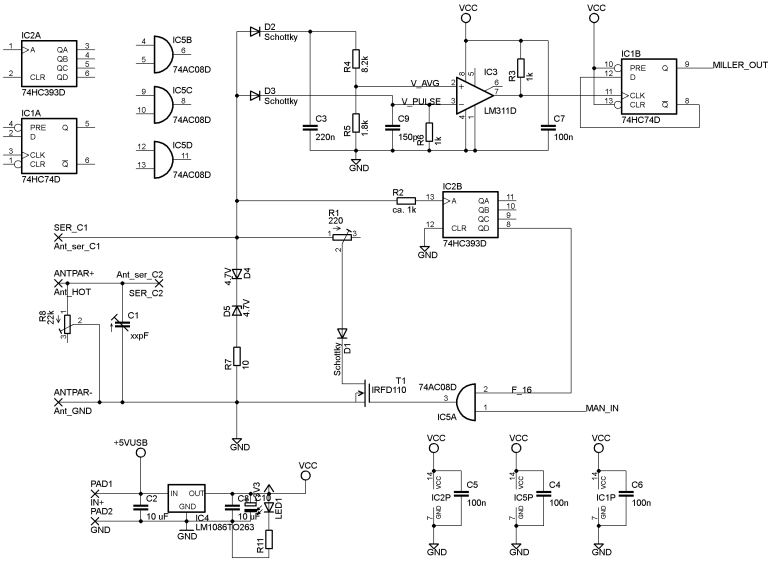

Fig. 4. Schematics of the power supply and the (de)modulation circuitry.