When Reverse-Engineering Meets Side-Channel Analysis – Digital Lockpicking in Practice

David Oswald(✉), Daehyun Strobel, Falk Schellenberg, Timo Kasper, and Christof Paar

Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Bochum, Germany
{david.oswald,daehyun.strobel,falk.schellenberg,timo.Kasper, christof.paar}@rub.de

Abstract. In the past years, various electronic access control systems have been found to be insecure. In consequence, attacks have emerged that permit unauthorized access to secured objects. One of the few remaining, allegedly secure digital locking systems—the system 3060 manufactured and marketed by SimonsVoss—is employed in numerous objects worldwide. Following the trend to analyze the susceptibility of real-world products towards implementation attacks, we illustrate our approach to understand the unknown embedded system and its components. Detailed investigations are performed in a step-by-step process, including the analysis of the communication between transponder and lock, reverse-engineering of the hardware, bypassing the read-out protection of a microcontroller, and reverse-engineering the extracted program code. Piecing all parts together, the security mechanisms of the system can be completely circumvented by means of implementation attacks. We present an EM side-channel attack for extracting the secret system key from a door lock. This ultimately gives access to all doors of an entire installation. Our technique targets a proprietary function (used in combination with a DES for key derivation), probably originally implemented as an obscurity-based countermeasure to prevent attacks.

Keywords: Access control · Symmetric key cryptosystem · Digital lock · Wireless door openers · EM side-channel attack · Obscurity

1 Introduction

Electronic access control systems are becoming increasingly popular and are on the way to replace conventional mechanical locks and keys in many applications. Often, the security of these systems is based on keeping the proprietary protocols and cryptographic algorithms secret. Admittedly, although violating Kerckhoffs's principle, this approach often prevents both mathematical analysis and implementation attacks, as long as the details remain undisclosed. In this paper, we focus on the latest generation "G2" of the widespread digital locking and access control system 3060 manufactured by SimonsVoss Technologies AG.

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 571–588, 2014.
DOI: 10.1007/978-3-662-43414-7_29,

A transponder, the digital equivalent of a mechanical key, wirelessly communicates with an electronically enhanced locking cylinder that—after successful authentication—mechanically connects the otherwise freewheeling knob to the bolt so that the door can be opened. By the example of this system, we illustrate which major problems have to be overcome in the real-world before successful (mathematical and) side-channel attacks can be mounted.

SimonsVoss is the European leader in digital locking and access control systems [18] and has sold over three million transponders for more than one million electronic locks. Despite the high price of the digital locks—compared to their mechanical counterparts—the purchase can pay out due to the flexible administration of access permissions via a wireless link, especially in buildings with a lot of doors and users.

1.1 Related Work

In the context of conventional mechanical locks exist well-known techniques to duplicate individual keys, if physical access is given. Likewise, the security of mechanical pin tumbler locks can be bypassed, e.g., by means of lockpicking with special tools. The practical threat of these attacks is usually negligible, as long as each key and each lock has to be treated individually. When it comes to the security of large installations and the corresponding master keys, even in the mechanical world the impact becomes more severe: In [3], a procedure is described that allows the creation of a master key to open all doors in an installation, if access to a single master-keyed lock and an associated key is given.

For wireless, electronic access control systems, obtaining the ingredients required for copying a transponder can be even more straightforward. They could be, e.g., eavesdropped from the Radio Frequency (RF) interface. To counteract attacks, numerous manufacturers developed and offer (often proprietary) cryptographic solutions. Most of these have in common that they turn out to be insecure from a cryptanalytical perspective once the secret protocols and algorithms have been reverse-engineered. Today most security products used for access control, such as Texas Instrument's Digital Signature Transponder (DST) [5], NXP's Mifare Classic cards [7,10], Hitag 2 transponders [21], and Legic Prime [17], have something in common: Mathematical attacks emerging from cryptographic weaknesses enable to break their protection in minutes.

For systems with a higher level of mathematical security, several examples of real-world side-channel attacks have demonstrated a huge attack potential: The 112-bit secret key of the Mifare DESfire MF3ICD40 smartcard (based on the 3DES cipher) can be extracted with Electro-Magnetic (EM)-based Side Channel Analysis (SCA) [16]. Likewise, the mathematical weaknesses found in the proprietary remote keyless entry system KeeLoq [1,4] are insufficient for a practical attack. However, a side-channel attack yielding the master key of the system allows to duplicate remote controls by one-time eavesdropping from several hundred meters [8].

The mathematical properties of the SimonsVoss digital lock and access control system 3060 "G2" are investigated in [20]. Sophisticated mathematical attacks, exploiting weaknesses of the proprietary obfuscation function (see Sect. 3) and a security vulnerability of the authentication protocol, are presented. As a consequence of the traditional cryptanalysis, transponders with a known identifier can be duplicated in seconds. The attack exploits that an internal protocol value is re-used as a "random number" in the next protocol run and thus can be obtained by an adversary. However, this flaw can be easily fixed, e. g., by a firmware update of the door lock, rendering the most severe mathematical attacks not applicable. The manufacturer SimonsVoss is currently preparing an according patch to protect existing installations against the cryptanalytical attacks.

1.2 Contribution

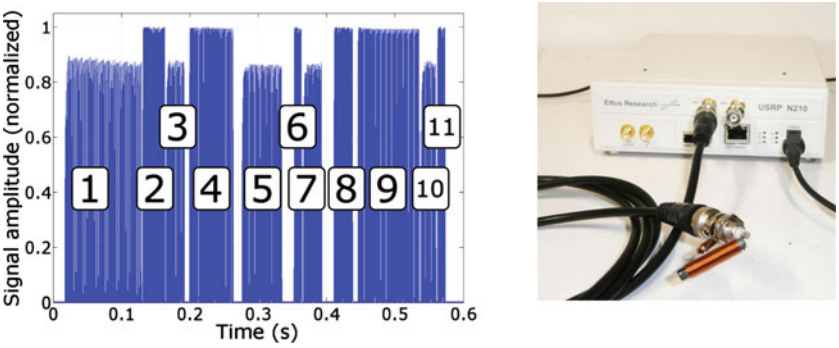
In contrast to the existing mathematical analyses, we illustrate how to circumvent the security mechanisms of SimonsVoss's system 3060 "G2" solely by means of physical attacks. In Sect. 2 we describe our varyingly successful attempts to bring the internals of the access control system to light and demonstrate from the perspective of an adversary how—despite the admittedly very time-consuming and demanding black-box analysis—a skilled engineer can finally succeed in extracting SimonsVoss's undisclosed secrets, detailed in Sect. 3. As the main part of our contribution in Sect. 4 we present a side-channel attack targeting the key derivation function running on a lock. Our non-invasive attack is able to extract the system key with approximately 150 EM measurements and enables access to all doors of an entire SimonsVoss installation. We finally discuss the learned lessons for SCA in the context of obscurity.

2 Reverse-Engineering: An Obstacle Course

Using the information SimonsVoss provides publicly, it is impossible to reason about the claimed level of security for the system 3060. Thus, in the following section, we present the process of reverse-engineering the inner workings of the digital locking cylinder and the corresponding transponder.

2.1 Radio Protocol

The documentation available on the Internet [19] yields only little information on the RF interface used to open a door: The transponder communicates with the counterpart in the door at 25 kHz at a specified maximum range of approx. 0.5 m. To capture the messages exchanged during an authentication, we used a simple coil made from copper wire and connected it to a USRP2 Software-Defined Radio (SDR) [9], as shown in Fig. 1b. We eavesdropped several (successful) protocol runs between a door and a transponder. An example of a monitored RF signal is depicted in Fig. 1a: The parts with the lower amplitude (marked 1, 3, 5, 7, and



(a) Protocol (b) USRP

Fig. 1. A successful run of the 11-step authentication protocol between transponder and door, recorded with the USRP and a connected antenna coil.

10) are sent by the transponder, while the other parts originate from the door (marked as 2, 4, 6, 8, 9, and 11).

From studying and comparing various recorded messages, some conclusions can be drawn: The used modulation scheme is On-Off Keying (OOK), i.e., the carrier at 25 kHz is switched on and off according to the sequence of data bits. The protocol starts with the transponder transmitting a synchronization preamble consisting of approximately 110 one-zero cycles. The raw bits of each message are encoded using a differential Manchester code [23], i.e., a transition during the bit period corresponds to a logical zero and no transition to a logical one. On the packet level, the messages follow a simple format: Data bytes following an 8-bit header `(0x7F)` have no further format requirements. In contrast, the 16-bit header `(0xFF, 0x7F)` indicates that an integrity check value, computed over the data bytes, has to be appended. This check value is computed by taking the complement of the bit-wise XOR of all data bytes.

We conducted several experiments that involved replaying previously recorded signals using the USRP2. However, the module in the door did not accept replayed messages and the door could not be unlocked. During further analyses of the interchanged messages, we found that some appear to be (pseudo-)random in each protocol run and thus that a cryptographic authentication scheme might be implemented. However, after several weeks of analyzing the RF interface we were convinced that some quite sophisticated scheme had been implemented by SimonsVoss. Unfortunately, the significance of the messages remained unclear. Thus, we decided to continue with the reverse-engineering on the hardware level.

2.2 Hardware and Circuit Boards

In order to understand the interaction between the mechanical and the electrical parts, we disassembled a door lock in a destructive manner, cf. Fig. 2a. We identified a magnetic pin that mechanically interlocks the bolt when voltage

(a) Door Hardware (b) Lock (c) Transponder

Fig. 2. "Exploded view" of the mechanical parts of a door lock 3061 and the PCBs contained in the door knob and a transponder 3064.

is applied to its contacts. The circuitry in the door thus controls whether the otherwise free-wheeling door knob is mechanically connected to the door latch and the deadbolt in order to grant access. No additional electronic circuits were found inside the lock, thus all security functionality must be implemented on the door's Printed Circuit Board (PCB). We observed several installations in which this circuitry is installed in the publicly accessible outer knob of the door. This holds especially for "panic"-locks which can be unlocked from the inside.

The PCB depicted in Fig. 2b is located inside one of the door knobs of the lock, that can be easily opened with a commercially available tool. The IC in the bottom-left corner is a proprietary SimonsVoss ASIC. On the right, a Microchip PIC16F886 microcontroller (μC) [15] is located. One connection between the `MCLR` pin of the PIC and the ASIC enables the former to wake up the μC from the power save mode. The ASIC is also connected to the μC's `OSC1/CLKIN`, hence the μC can be clocked from the ASIC. Moreover, an external EEPROM in an SOT-8 package is connected to the μC. The plastic case of the transponder contains the PCB depicted in Fig. 2c. A comparison with the lock shows many identical parts, e.g., a PIC16F886 connected to an external EEPROM and the ASIC. A push-button triggering an authentication to the door is connected to the PIC.

The ASIC is a 16-pin IC with a custom label ("`MA124D`") for which we could find no information on the Internet. The pinout and the activity on the data lines did not point to any distinct manufacturer, thus, we decapsulated the silicon die of several ICs, cf. [2, p.10]. Figure 8 in Appendix B depicts a high resolution image of the silicon die. After analyzing microscopic pictures of the chip, it turned out that it contains a mask-configurable gate array IC with a rather limited amount of implementable logic. Besides, the ASIC does not feature internal memories. We figured out that the majority of the available logic cells is used (i) for a counter that periodically wakes up the μC and (ii) for functions of the RF interface. It became clear that the ASIC does not contain security-related functions and thus can be ruled out as a target for further analyses: All relevant algorithms must thus be contained in the PIC μC.

As an (almost) last resort in order to understand the "randomly" looking protocol and find a security vulnerability in the system, we proceeded with

hypothesizing that a standard cipher (e.g., AES or DES) might be implemented on the PIC. Assuming that some of the varying bytes in the protocol could be input or output of a one of these ciphers, we tapped the power line of the μC and spent some more weeks trying to recover cryptographic keys, by acquiring power traces and performing a Correlation Power Analysis (CPA) for the guessed ciphers, one after another. However, none of our numerous attempts turned out to be successful. After analyzing the radio protocol, the circuit boards, the proprietary ASIC, and trying "shot-in-the-dark" power-analysis attacks, we still had no clue about the security features of the SimonsVoss system 3060 and were close to giving up with the conclusion that it was "secure enough". However, as a last attempt, we decided to try to obtain the machine code of the PIC.

2.3 The Breakthrough: Extracting and Reverse-Engineering the Firmware of the PIC

After connecting a PIC programmer to the μC and trying to read out its content it turned out that the read-out protection was enabled. Thus, following the methods proposed in [24] and [13], we tried to erase the code and data read-out protection bits: The PIC was decapsulated, and the memory cells containing the protection bits were exposed to Ultraviolet-C (UV-C) light. Even though Microchip covers the top of the respective cells of the PIC16F886 with small metal plates as a countermeasure, applying the UV-C light at an angle (so that it bounces off multiple structures around the floating gate) deactivated the read-out protection. The whole process required less than 30 min. After that, the complete content of the PIC's program memory and its internal EEPROM could be extracted. We performed the read-out process for the PICs of several transponders and door locks and started to analyze their program code.

In order to disassemble and understand the extracted program code, we utilized the reverse-engineering tool IDA Pro [12]. Analyzing the program code, we were able to recover most previously unknown details of the SimonsVoss system 3060, including the authentication protocol and the employed cryptographic primitives, cf. Sect. 3. In addition to performing a static analysis of the program code, we also inserted a debug routine `debug_dump` into the assembly code and re-programmed the PICs with our modified firmware. Given that the correct (matching) combination of program code and internal and external EEPROM content is written, the modified transponder or door lock operates correctly. The inserted routine allows to dump the registers and the SRAM during the execution of the original program. To this end, we utilized an unused pin of the μC to transfer the memory dump in a straightforward serial manner.

The debug routine cannot be directly inserted into the program code, as this would lead to a shift of all (absolute) jump addresses and thus render the program inoperable. To solve this problem, we had to overwrite code that was found to be not linked to the authentication protocol with the code of `debug_dump`. Since the PIC16F886 uses a segmented program memory (consisting of four banks with 2 K instruction words each), we furthermore had to place a small piece of wrapper code at the end of each segment (thereby overwriting a few

unused instructions). Then, the call to `debug_dump` was inserted as follows: A target instruction (e.g., a `movwf`) is replaced by a call to the wrapper function. The wrapper function then selects the segment in which `debug_dump` resides and subsequently invokes `debug_dump`. Before outputting the dump, the value of the `STATUS` register is saved in an unused memory location to preserve the state of the actual program. Before returning, this value is restored, and the subsequent instructions (of the wrapper) do not modify the `STATUS` register. After `debug_dump` has returned to the wrapper, the segment is reset to the originally selected value. Then, the wrapper executes the replaced instruction before returning to the normal program flow.

With the capability to dump the memory contents, e.g., during the execution of a successful authentication protocol run, we are able to verify the results of the static analysis and to understand also those parts of the code that heavily depend on external input (e.g., from received data or data stored in the external memory). Obtaining and analyzing the program code was the essential step to enable a detailed scrutiny, including mathematical and side-channel analyses. Being now able to understand the exchanged messages and the used cryptographic functions, the authentication mechanism described in Sect. 3 can be attacked in several ways, including the SCA presented in Sect. 4. Without the complete reverse-engineering, the SCA of the—in this regard extremely vulnerable—PIC μC would have been impossible.

3 SimonsVoss's Proprietary Cryptography

In this section, as a mandatory prerequisite for an SCA, we summarize the relevant results of reverse-engineering the code running on the PIC of the transponder and the lock. To this end, we describe the key derivation mechanism, the cryptographic primitives, and the protocol used for mutual authentication.

The authentication protocol consists of in total eleven steps that are given in Fig. 7 in Appendix A. In the symmetric-key scheme, the transponder and the lock prove that they know a shared long-term secret K_T. On each transponder, this individual 128-bit key K_T is computed as the XOR of a 128-bit value $K_{T,\text{int}}$ stored in the internal EEPROM of the μC (not accessible from the outside) and a 128-bit value $K_{T,\text{ext}}$ stored in the (unprotected) external EEPROM, i.e., $K_T = K_{T,\text{ext}} \oplus K_{T,\text{int}}$. Each door within an entire SimonsVoss installation has an identical set of four 128-bit keys $K_{L,1}$, $K_{L,2}$, $K_{L,3}$, $K_{L,4}$, here called *system key*. Again, these keys are computed as the XOR of values contained in the internal and in the external EEPROM. However, the internally stored value is identical for all four keys, i.e., $K_{L,j} = K_{L,j,\text{ext}} \oplus K_{L,\text{int}}$. Note that when one of the four $K_{L,j}$ has been recovered, the remaining three can be determined after reading the respective values from the external EEPROM.

The system key is used to derive the key K_T of a transponder to be authenticated. After receiving the identifier I_T of a transponder, the lock computes K_T on-the-fly using a key derivation function $\mathcal{K}$ involving the system key, previously exchanged "authentication data" D, and I_T. The authentication data D

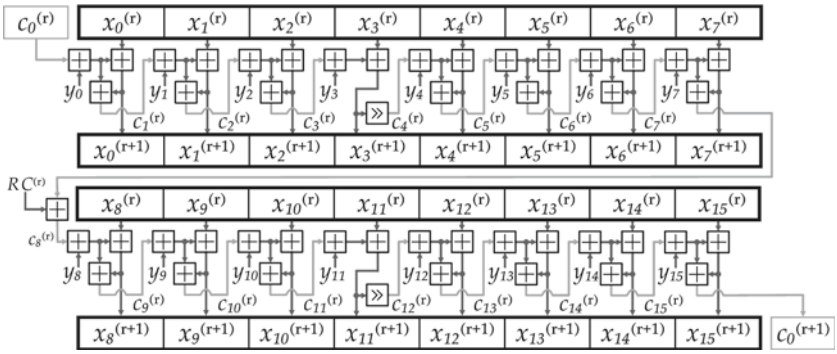


Fig. 3. One round r of the obscurity function $\mathcal{O}$: The key bytes y_i are constant, while the ${x_i}^{(r)}$ and the chaining values ${c_i}^{(r)}$ are updated in each round. The first chaining value is ${c_0}^{(1)} = RC^{(0)}$. The two 8-byte halves $x_0 \dots x_7$ and $x_8 \dots x_{15}$ are processed in an identical manner.

is a value sent by the transponder and controls certain protocol functions. D is, however, transmitted in plain and unrelated to the cryptographic security of the protocol. All valid transponder identifiers that can authenticate to a particular door lock are stored unencrypted in its external EEPROM. Note that the key derivation is always executed by the door lock, even if it does not know the received I_T.

The key derivation $\mathcal{K}$ consists of two building blocks: a modified DES denoted as $\mathcal{D}$ and a SimonsVoss-proprietary function $\mathcal{O}$ which we refer to as "obscurity function". $\mathcal{D}(K;\, M)$ is a slightly modified DES encrypting a 64-bit plaintext M under a 64-bit key K.

$\mathcal{O}(Y;\, X)$ consists of eight rounds r, with $1 \leq r \leq 8$. It takes a 16-byte plaintext $X = ({x_0}^{(1)} \dots {x_{15}}^{(1)})$ and a 16-byte key $Y = (y_0 \dots y_{15})$ to compute a 16-byte ciphertext ${x_0}^{(9)} \dots {x_{15}}^{(9)}$. Figure 3 depicts the structure of one round of $\mathcal{O}$. The internal "chaining values" ${c_1}^{(r)} \dots {c_{15}}^{(r)}$ are processed horizontally in each round r and the last chaining value is used as the input ${c_0}^{(r+1)}$ for the subsequent round $r + 1$. The round constants $RC^{(r)}$ are fixed byte values, i.e., they are identical for every execution of $\mathcal{O}$, that are added in each round after the first eight bytes have been processed. The first chaining value is initialized with ${c_0}^{(1)} = RC^{(0)}$. All additions and shifts are performed modulo 256.

$\mathcal{K}$ takes the system key and a 128-bit parameter P_0 as inputs, where P_0 is derived from the first three bytes of I_T and the first three bytes of the authentication data D as $P_0 = (I_{T,0}, I_{T,1}, I_{T,2} \,\&\, \texttt{0xC7}, D_0, D_1, D_2 \,\&\, \texttt{0x3F}, 0, \dots, 0)$. The transponder key K_T is then computed by a series connection of $\mathcal{O}$, $\mathcal{D}$, and $\mathcal{O}$ as

$$\mathcal{K}(K_{L,j};\, P_0) = \mathcal{O}\left(P_0; \mathcal{D}\left(\mathcal{O}\left(K_{L,j}; P_0\right)_{64..127}; \mathcal{O}\left(K_{L,j}; P_0\right)_{0..63}\right) || 0 \dots 0\right).$$

Depending on the two Most Significant Bits (MSBs) of $I_{T,2}$, one of the four $K_{L,j}$ is selected as the key for the first (innermost) instance of $\mathcal{O}$ in $\mathcal{K}$ to encrypt P_0. The result is split into two 64-bit halves, the lower 64 bit being the plaintext

and the upper 64 bit being the key of $\mathcal{D}$. The 64-bit result of $\mathcal{D}$ is then padded with 64 zero bits and encrypted with $\mathcal{O}$, this time using P_0 as the key. The resulting ciphertext is the transponder key K_T used in the subsequent steps of the challenge-response protocol. A lock is opened, if K_T on the transponder and K_T derived by the door match. To this end, both transponder and door compute a 64-bit response involving the challenge C and several protocol values as

$$\mathcal{R}\left(K_T; P_1, P_2\right) = \mathcal{D}\left(\mathcal{O}\left(\mathcal{O}(K_T; P_1); P_2\right)_{64..127}; \mathcal{O}\left(\mathcal{O}\left(K_T; P_1\right); P_2\right)_{0..63}\right),$$

with

$$P_1 = (C_0, \ldots, C_{10}, D_6, \ldots, D_9, 0)$$
$$P_2 = (I_{L,2}, I_{T,2}, I_{T,3}, D_3, D_4, D_5, 0, \ldots)$$

The transponder sends the first 32-bit half R_0 of the output of $\mathcal{R}$, to which the door responds with the second half R_1, if R_0 was correct. Obviously, the key derivation function $\mathcal{K}$ is the main target for an SCA, because recovering the system key allows to derive the key of any transponder in an installation given its I_T.

4 Extraction of the System Key with SCA

In this section, we describe the steps to perform a side-channel attack on the Device-Under-Test (DUT), the SimonsVoss door lock 3061. As the main result, we show that the system key can be extracted from the employed PIC μC with a non-invasive, CPA-based attack using approximately 150 traces. Possessing the system key, an adversary is able to create functionally identical clones of *all* transponders in an entire SimonsVoss installation. Note that the SCA can in general also be applied to a transponder, e.g., in order to duplicate it, but in a practical setting it is highly unlikely that an adversary will take the efforts for the attack just for cloning a single transponder.

4.1 Side-Channel Attacks

A side-channel attack is usually performed in two steps. First, the adversary has physical access to the target device and acquires a side-channel signal (e.g., the power consumption or the EM emanation) during the cryptographic computation. This is repeated N times with different input data M_n, yielding N time-discrete waveforms $w_n(t)$ or traces. To recover the cryptographic key, the traces are then statistically processed in the evaluation phase, e.g., using the Pearson correlation coefficient when performing a CPA [6]. The adversary fixes a (small) subset $\mathcal{K}_{cand} \subseteq \mathcal{K}$ (e.g., the 256 possible 8-bit subkeys entering one S-box of the AES) and considers all key candidates $k \in \mathcal{K}_{cand}$. Then, for each $k \in \mathcal{K}_{cand}$ and for each $n \in \{0, \ldots, N-1\}$, a hypothesis $V_{k,n}$ on the value of some intermediate (e.g., the output of one 8-bit AES S-box) is computed. Using a power model f, this value is then mapped to $h_{k,n} = f(V_{k,n})$ to describe the process that causes the side-channel leakage. In practice, a Hamming weight (HW) or

Hamming distance (HD) power model is often suitable for CMOS devices like μCs [14]. In order to detect the dependency between $h_{k,n}$ and $w_n(t)$, the correlation coefficient $\rho_k(t)$ (for each point in time t and each key candidate $k \in \mathcal{K}_{cand}$) is given as

$$\rho_k(t) = \frac{cov(w(t), h_k)}{\sqrt{var(w(t))\, var(h_k)}}$$

with $var(\cdot)$ indicating the sample variance and $cov(\cdot,\cdot)$ the sample covariance according to the standard definitions [22]. The key candidate $\hat{k}$ with the maximum correlation $\hat{k} = \arg\max_{k,t} \rho_k(t)$ is assumed to be the correct secret key. When for instance attacking an implementation of the AES, this process is performed for each S-box separately, yielding the full 128-bit key with a much lower complexity of $16 \cdot 2^8$ compared to 2^{128} steps for an exhaustive search. However, for the obscurity function $\mathcal{O}$ used in the key derivation step (cf. Sect. 3), performing a side-channel attack is more complex. Hence, in the following Sect. 4.2, the process for extracting the system key by means of SCA is described in detail.

4.2 Theoretical Attack: Predicting Intermediate Values in $\mathcal{O}$

The most promising target for an SCA to recover the system key is the first instance of $\mathcal{O}$ in the key derivation. In the following, we (theoretically) derive an attack to obtain the full 16-byte key $y_0 \ldots y_{15}$.

Note that only the first six bytes of the input to $\mathcal{O}$ can be chosen by the adversary, since the remaining bytes of P_0 are set to zero. Due to the carry propagation properties of $\mathcal{O}$ (which are described in detail in [20]), these input bytes do not lead to a "full" randomization of all intermediates in the first round of the obscurity function. A straightforward CPA of the first round targeting the addition operation ${x_i}^{(\mathrm{r}+1)} = {x_i}^{(\mathrm{r})} + {c_i}^{(\mathrm{r})} + y_i$ hence only allows to recover the first seven key bytes $y_0 \ldots y_6$.

For the remaining key bytes, at least a part of the addition in the first round is carried out with completely constant data, ruling out a CPA: For revealing y_7, one obtains already two candidates, because the Least Significant Bit (LSB) of ${c_7}^{(1)}$ does not depend on the varying part of the input and hence remains constant. For ${c_8}^{(1)}$, two bits are not randomized, leading to four candidates for y_8 (if c_7 was known). Thus, to obtain these two bytes, two CPAs each with four additional candidates would be necessary. To recover all key bytes using this approach, an overall number of $2^{1+2+3+4+5+4+5+6+7} = 2^{37}$ key candidates would have to be tested, leading to an impractical attack. Hence, we utilize further properties of $\mathcal{O}$ to reduce the computational cost by extending the attack to initially non-recoverable key bytes in subsequent rounds of $\mathcal{O}$.

First, note that all bits in the (initially not fully randomized) update involving the key bytes $y_7 \ldots y_{15}$ are fully dependent on ${x_0}^{(1)} \ldots {x_5}^{(1)}$ after two, three, four, five, six, six, six, seven, and seven rounds, respectively. Thus, these key bytes can be recovered by means of a CPA in the respective round, if all other values preceding the update operation for a targeted key byte y_i can be simulated. Assuming that all key bytes up to y_i and all ${c_0}^{(\mathrm{r})}$ up to the respective

round are known, the only unknown constant in the i-th update operation is y_i. The correct value can be determined using a CPA with 256 candidates for y_i.

The remaining problem with this attack is how to determine ${c_0}^{(\mathrm{r})}$. For this, note that in the second round, ${c_0}^{(2)}$ is independent of ${x_0}^{(1)} \ldots {x_5}^{(1)}$, i.e., a constant only depending on the (constant) key bytes y_i. Hence, ${c_0}^{(2)}$ can be found using a CPA, because y_0 was already determined in the first round, so ${x_0}^{(2)}$ can be computed. The CPA for obtaining ${c_0}^{(2)}$ is performed with 256 candidates. Having found ${c_0}^{(2)}$, all values up to the update of ${x_7}^{(2)}$ can be computed. This update, in turn, only depends on known values (${x_7}^{(1)}$, ${c_7}^{(1)}$, and ${c_7}^{(2)}$) and the unknown key byte y_7. Hence, with 256 candidates, y_7 can be determined with a CPA.

In the third, fourth, and fifth round, only the MSBs of ${c_0}^{(3,4,5)}$ vary. The remaining bits are constant and can be recovered. In addition, due to the multiplication by two (i.e., a binary left-shift) in the propagation of c_i, the unknown MSB only affects (the MSB of) ${x_0}^{(4)}$, ${x_0}^{(5)}$, ${x_1}^{(5)}$, ${x_0}^{(6)}$, ${x_1}^{(6)}$, and ${x_2}^{(6)}$. Subsequent bytes do not depend on the unknown MSB and can be fully predicted. Hence, it is possible to recover y_8, y_9, and y_{10} in rounds three to five. Finally, in the sixth, seventh, and eighth round, the three MSBs of ${c_0}^{(6)}$ and the five MSBs of ${c_0}^{(7)}$ and ${c_0}^{(8)}$ vary. Again, the constant part of ${c_0}^{(6,7,8)}$ can still be recovered. For the varying three MSBs, only ${x_0}^{(7)} \ldots {x_2}^{(7)}$ are affected. This recoverable part is sufficient to fully predict the state update for the targeted key bytes y_{11}, y_{12}, and y_{13}. Finally, for round seven, the change in the five MSBs of ${c_0}^{(7)}$ only affects ${x_0}^{(8)} \ldots {x_6}^{(8)}$, posing no problem to the recovery of the remaining bytes y_{14} and y_{15}.

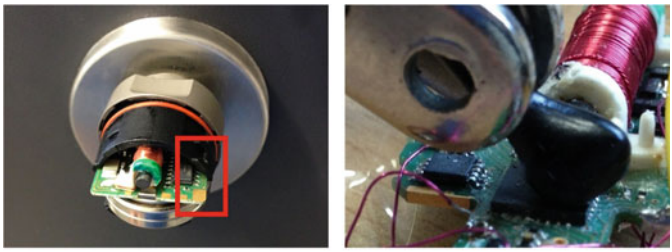
In short, the attack can be summarized as follows: First, one recovers the first seven key bytes in round one. Then, the constant part of c_0 at the beginning of each round has to be found. Finally, all key bytes for which the update operation is fully randomized in the respective round can be obtained.

4.3 Practical Results

In the following, we first describe the measurement setup used to acquire side-channel traces for the SimonsVoss door part 3061. Using techniques and results of the reverse-engineering described in Sect. 2, we profile the DUT and determine the point in time where the leakage occurs in the power trace. Finally, applying the attack described in Sect. 4.2 to real-world power traces, we recover the system key using a limited number of power traces in a non-invasive manner.

To trigger the key derivation on the DUT, we directly connect to the data lines between the ASIC and the PIC. The respective pins (marked with a red rectangle in Fig. 4a) are accessible without removing the PCB from the door. Equivalently, the communication with the DUT could also be performed sending data over the RF interface, e.g., using the USRP2 as described in Sect. 2.1. We primarily decided not to use the RF interface to increase the (mechanical) stability of the measurement setup.

We non-invasively measured the power consumption during the execution of the key derivation function for randomly chosen ID I_T and authentication data



(a) PCB in the door part (b) EM probe on PIC

Fig. 4. Setup and EM trace for SCA of the door part

D (apart from a few bits that are required to be set by the protocol). To this end, we placed an EM probe on the package close to the power supply pins of the PIC, cf. Fig. 4b. Initial experiments to measure the power consumption by inserting a shunt resistor into the battery connection yielded heavily smoothed power traces, presumably due to several bypass/voltage stabilization capacitors on the PCB. Moreover, the ASIC is also connected to the main battery supply, resulting in wide-band, high-amplitude noise that could not be filtered out.

In contrast, the EM probe at the given position mainly picked up the power consumption of the PIC. Using the described setup, we recorded 1,000 power traces using a Digital Storage Oscilloscope (DSO) at a sample rate of 500 MHz. With the current measurement setup, due to delays caused by the protocol, about 10 traces can be acquired per minute. Note that the PIC runs on the internal RC oscillator at a frequency of approximately 8 MHz. Hence, we further (digitally) bandpass-filtered the recorded traces. Experimentally varying the lower and upper frequency of the passband, we found that a passband from 6 MHz to 9 MHz yields the best result.

To determine the relevant part of the power trace belonging to the key derivation, we initially inserted a small function into the (otherwise unmodified) code of the PIC using the method described in Sect. 2.3. This function generates a rising edge on an unused pin of the PIC, serving as a trigger signal for profiling purposes. Figure 5 exemplarily depicts the part of a trace belonging to the initial execution of $\mathcal{O}$ in the key derivation. The eight rounds of the function can be recognized as eight distinct "humps". Furthermore, a unique pattern occurs at the beginning of the relevant part (and each round). This pattern can serve for alignment purposes: Having profiled the DUT, we removed the artificial trigger signal and recorded traces for the original, unmodified code. We then used the pattern to align the traces for the actual CPA attack.

Having determined the relevant part of the trace and the appropriate preprocessing steps, we practically performed the (theoretical) attack described in Sect. 4.2. We use the HD between a byte ${x_i}^{(\mathrm{r})}$ and its updated value ${x_i}^{(\mathrm{r}+1)}$ as the power model, i.e., $h = \mathrm{HD}\left({x_i}^{(\mathrm{r})}, {x_i}^{(\mathrm{r}+1)}\right)$ (dropping the indices n for the trace and k for the key candidate for better readability). This model was derived based on the analysis of the assembly code implementing $\mathcal{O}$ and the leakage model

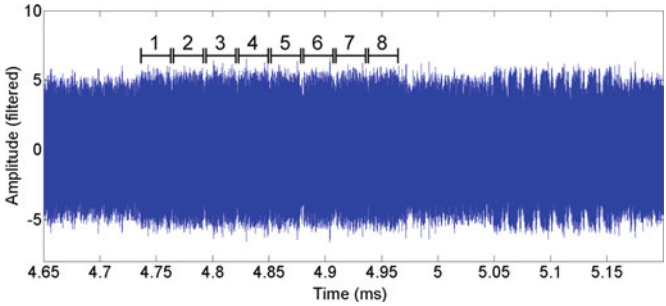
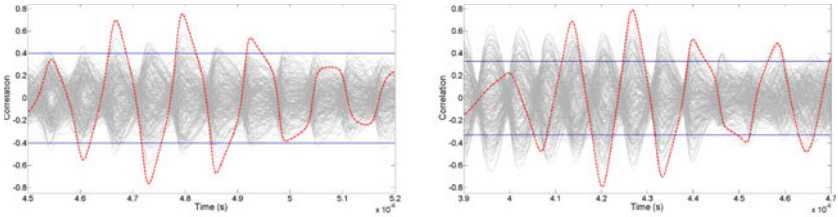


Fig. 5. Filtered EM trace during the execution of the first obscurity function in the key derivation



(a) y_7 in Round 2 (b) ${c_0}^{(2)}$ in Round 2

Fig. 6. Correlation for key byte y_7 after 100 traces and ${c_0}^{(2)}$ after 150 traces. Correct candidate: red, dashed

for the PIC series described in [11]. Using this model, we obtained correlation values of approximately 0.75 for the correct candidate, while all other (wrong) candidates exhibit a lower value. This allows to unambiguously determine the key with approximately 100 traces, as exemplarily depicted in Fig. 6a.

The CPA for ${c_0}^{(\mathrm{r})}$ exhibit a similar behavior, cf. for instance Fig. 6b for ${c_0}^{(2)}$. However, for determining the partial ${c_0}^{(\mathrm{r})}$ in later rounds, a (slightly) higher number of traces is needed (especially for round six and seven), since only the LSBs of the respective intermediate values are predicted correctly. Still, approximately 150 traces are sufficient to obtain the maximum correlation at the correct point in time for the correct candidate.

Note that—at a higher computational cost—the CPA on ${c_0}^{(6)}$ and ${c_0}^{(7)}$ could be left out altogether: The attack on the actual key bytes in round six and seven could be carried out for all 2^5 or 2^3 candidates for ${c_0}^{(6)}$ or ${c_0}^{(7)}$, respectively. This would increase the number of candidates to $2^5 \cdot 2^8 = 2^{13}$ and $2^3 \cdot 2^8 = 2^{11}$. For this amount of candidates, a CPA can still be executed efficiently.

5 Conclusion

In this paper, we presented a practical, non-invasive side-channel attack on the SimonsVoss 3060 system, allowing to extract the system key from a door lock in an installation. The attack requires approximately 150 EM traces which can be recorded in approximately 15 min using our current measurement setup. An adversary possessing the system key ultimately gains access to all doors of the entire installation.

Reverse-engineering the embedded code of the PIC μC enabled the profiling and the development of a suitable prediction function for the CPA. Surprisingly, the cryptanalytical weakness of the employed obscurity function (highly linear structure, slow avalanche effect) together with its specific usage in the key derivation (input largely constant) require a more complicated SCA compared to attacking a standard cipher, e.g., DES or AES.

Note that the mathematical attacks of [20] can be fixed with a firmware update that is currently being prepared by SimonsVoss. In contrast, preventing the SCA without replacing all hardware components (locks and transponders) is very difficult in our opinion. The program memory and the RAM of the PIC of the lock are almost fully utilized. This rules out the implementation of countermeasures, e.g., masking, requiring an (even slightly) increased program size or RAM usage. Randomizing the timing of the algorithms appears likely to be ineffective, because (i) clear patterns in the EM trace can be detected and (ii) no suitable entropy source is available. Furthermore, software countermeasures could again be reverse-engineered and closely inspected on a program-code level.

The most important lesson to be learned from the demonstrated attack is that the overall security of a system depends on the weakest link in the chain: The cryptographic algorithm used by SimonsVoss, i.e., a DES core combined with an obscurity function that obfuscates the input and output bytes of the DES and enlarges the key length to 128 bit, provided a reasonable level of security: As long as the proprietary scheme remained undisclosed, neither a brute-force attack nor mathematical cryptanalysis of the unknown cipher were a practical option. Likewise, an analysis of the protocol and SCA attacks targeting a hypothesized cipher were fruitless.

One single factor, however, overthrew the security of the SimonsVoss system: a vulnerability of the PIC microcontroller that enables to bypass the code readout protection. Without this weakest link of the chain and the respective central step of the analysis—reverse-engineering the program code—probably no feasible (mathematical and implementation) attacks would have been found.

A SimonsVoss Authentication Protocol

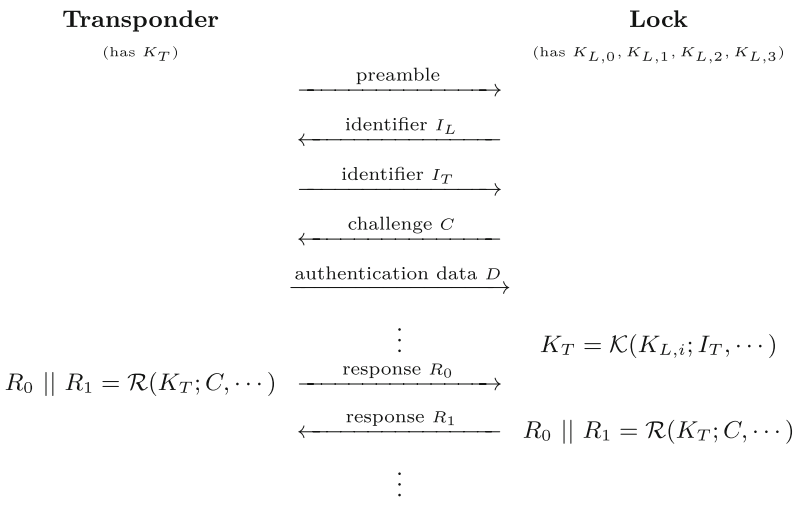


Fig. 7. Protocol for the mutual authentication between a transponder and a lock

B SimonsVoss Proprietary ASIC

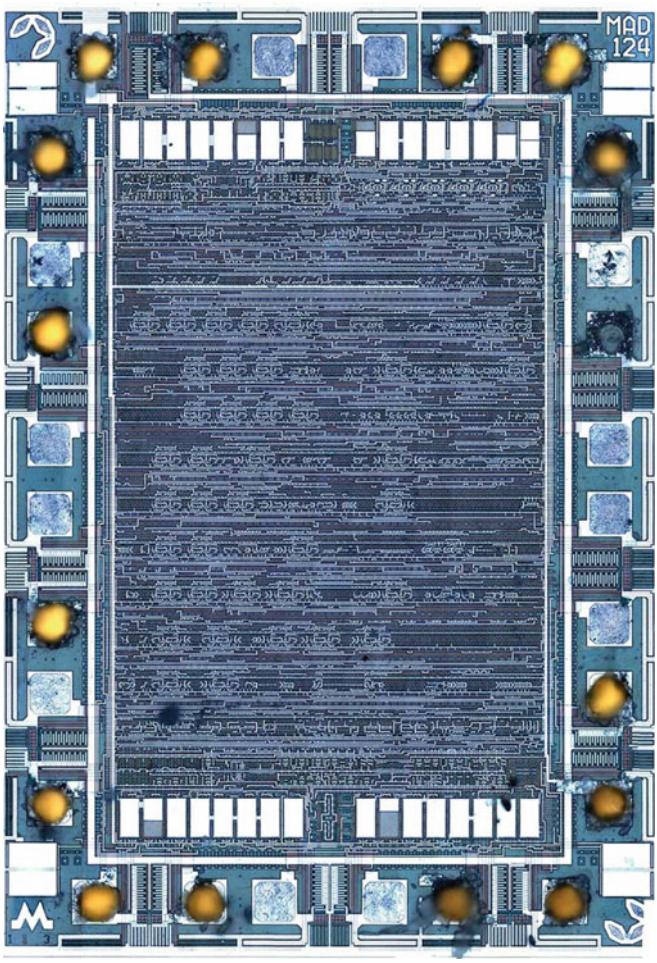


Fig. 8. Microscope photograph of the SimonsVoss ASIC

References

1. Aerts, W., Biham, E., De Moitié, D., De Mulder, E., Dunkelman, O., Indesteege, S., Keller, N., Preneel, B., Vandenbosch, G., Verbauwhede, I.: A Practical Attack on KeeLoq, pp. 1–22. Springer, New York (2010)
2. Beck, F.: Präparationstechniken Für Die Fehleranalyse an Integrierten Halbleiterschaltungen. VCH Verlagsgesellschaft, Weinheim (1988)

3. Blaze, M.: Rights amplification in master-keyed mechanical locks. IEEE Secur. Priv. **1**(2), 24–32 (2003). http://www.crypto.com/papers/mk.pdf
4. Bogdanov, A.: Attacks on the keeloq block cipher and authentication systems. In Workshop on RFID Security (RFIDSec'08) (2007). http://rfidsec07.etsit.uma.es/slides/papers/paper-22.pdf
5. Bono, S.C., Green, M., Stubblefield, A., Juels, A., Rubin, A.D., Szydlo, M.: Security analysis of a cryptographically-enabled RFID device. In: Proceedings of the 14th Conference on USENIX Security Symposium, vol. 14. USENIX Association (2005) http://www.usenix.org/events/sec05/tech/bono/bono.pdf
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
7. Courtois, N.: The dark side of security by obscurity - and cloning mifare classic rail and building passes, anywhere, anytime. In SECRYPT, pp. 331–338. INSTICC (2009)
8. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the power of power analysis in the real world: a complete break of the KeeLoq code hopping scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
9. Ettus Research. USRP N200/210 Networked Series (2012). https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf
10. Garcia, F.D., van Rossum, P., Verdult, R., Schreur, R.W.: Wirelessly pickpocketing a mifare classic card. In: IEEE Symposium on Security and Privacy, pp. 3–15. IEEE (2009)
11. Goldack, M.: Side-Channel Based Reverse Engineering for Microcontrollers. Diploma thesis, Ruhr-University Bochum (2008). https://www.emsec.rub.de/media/attachments/files/2012/10/da_goldack.pdf
12. Hex-Rays. IDA Starter Edition. http://www.hex-rays.com/products/ida/processors.shtml as of 24 July 2013
13. Huang, A.: Hacking the PIC 18F1320 (2005). http://www.bunniestudios.com/blog/?page_id=40 as of 24 July 2013
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer (2007). http://www.dpabook.org
15. Microchip Technology Inc. PIC16F882/883/884/886/887 Data Sheet (2009). http://ww1.microchip.com/downloads/en/devicedoc/41291f.pdf
16. Oswald, D., Paar, C.: Breaking mifare DESfire MF3ICD40: power analysis and templates in the real world. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)
17. Plötz, H., Nohl, K.: Legic Prime: Obscurity in Depth (2009). http://events.ccc.de/congress/2009/Fahrplan/attachments/1506_legic-slides.pdf
18. SimonsVoss Technologies AG. SimonsVoss posts record sales yet again in (2011). http://www.simons-voss.us/Record-sales-in-2011.1112.0.html?&L=6 as of 24 July 2013
19. SimonsVoss Technologies AG. Infocenter - Downloads (2006). http://www.simons-voss.com/Downloads.45.0.html?&L=1 as of 24 July 2013
20. Strobel, D., Driessen, B., Kasper, T., Leander, G., Oswald, D., Schellenberg, F., Paar, Ch.: Fuming acid and cryptanalysis: handy tools for overcoming a digital locking and access control system. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 147–164. Springer, Heidelberg (2013)

21. Verdult, R., Garcia, F.D., Balasch, J.: Gone in 360 seconds: Hijacking with Hitag2. In USENIX Security Symposium, pp. 237–252. USENIX Association, August 2012. https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final95.pdf
22. Eric W. Weisstein. Variance. Mathworld - A Wolfram Web Resource, December 2010. http://mathworld.wolfram.com/Variance.html
23. Wikipedia. Differential Manchester Encoding – Wikipedia, The Free Encyclopedia (2012). Accessed 12 November 2012
24. Zonenberg, A.: Microchip PIC12F683 teardown (2011). http://siliconexposed.blogspot.de/2011/03/microchip-pic12f683-teardown.html