INVITED PAPER

Microcontrollers as (In)Security Devices for Pervasive Computing Applications

This paper discusses possible threats to embedded systems using two microcontroller case studies.

By Daehyun Strobel, David Oswald, Bastian Richter, Falk Schellenberg, and Christof Paar, *Fellow IEEE*

ABSTRACT | Often overlooked, microcontrollers are the central component in embedded systems which drive the evolution toward the Internet of Things (IoT). They are small, easy to handle, low cost, and with myriads of pervasive applications. An increasing number of microcontroller-equipped systems are security and safety critical. In this tutorial, we take a critical look at the security aspects of today's microcontrollers. We demonstrate why the implementation of sensitive applications on a standard microcontroller can lead to severe security problems. To this end, we summarize various threats to microcontroller-based systems, including side-channel analysis and different methods for extracting embedded code. In two case studies, we demonstrate the relevance of these techniques in real-world applications: Both analyzed systems, a widely used digital locking system and the YubiKey 2 onetime password generator, turned out to be susceptible to attacks against the actual implementations, allowing an adversary to extract the cryptographic keys which, in turn, leads to a total collapse of the system security.

KEYWORDS | Code extraction; microcontroller; real-world attacks; reverse engineering; side-channel analysis

I. INTRODUCTION

A surprisingly large share of the world's central processing unit (CPU) market still goes to low-end, off-the-shelf 8-b μCs. They combine several versatile modules in a single chip, e.g., a processor unit, program and data memory, clock generation and peripherals, and are therefore often called a self-contained computing system. Corresponding embedded applications range from household appliances over car electronics to industrial control. A steadily increasing number of μC applications require security. Examples include contactless and traditional smart cards, electronic control units (ECUs) in cars, tokens for access control, industrial machine controllers (for services such as software download or feature activation), and even medical implants. The continuing trend toward the Internet of Things (IoT) will only increase the need for embedded applications with strong security features. It is thus of great interest for design engineers and users to assess the feasibility of μCs as security devices.

In this tutorial, we will discuss techniques to attack (cryptographic) implementations running on μCs. First, we introduce side-channel analysis (SCA) in Section II. SCA allows breaking mathematically secure ciphers by analyzing physical leakages observed during the execution on a field-programmable gate array (FPGA), an application-specific integrated circuit (ASIC), or a μC. SCA has its origin in the late 1990s, when Kocher, Jaffe, and Jun introduced the first attacks based on the timing behavior and power consumption of cryptographic implementations [1], [2]. It turns out that 8-b μCs are particularly susceptible against SCA. For example, extracting the key of an unprotected Advanced Encryption Standard (AES) implementation (AES is the world's most widely used

Manuscript received September 2, 2013; accepted May 1, 2014. Date of publication June 5, 2014; date of current version July 18, 2014. This work was supported in part by the German Federal Ministry of Education and Research BMBF under Grant 01IS10026A, Project EXSET.

The authors are with the Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum 44801, Germany (e-mail: daehyun.strobel@rub.de; david.oswald@rub.de; bastian.richter@rub.de; falk.schellenberg@rub.de; christof.paar@rub.de).

Digital Object Identifier: 10.1109/JPROC.2014.2325397

encryption algorithm) on an Atmel AVR ATmega8 is feasible with less than 50 power measurements under ideal conditions. Masking and hiding methods are possible countermeasures to complicate side-channel attacks, but often cannot prevent them completely [3].

The next topic of this tutorial is concerned with extracting and reverse engineering of embedded program code of a μC (cf. Section III), which can be a serious threat for some applications. Nearly all of today's μCs have some mechanism for code protection. In most cases, this is a lock bit that, if enabled, shall ensure that the firmware and data stored on the device cannot be read by an unauthorized party. This feature is used by numerous manufacturers to protect intellectual property and to assure safety and security. Of course, this assumes that the protection cannot be circumvented. However, research has shown that this assumption is not valid for several devices. For instance, Skorobogatov investigated several μCs with respect to their susceptibility against invasive, semi-invasive, and noninvasive attacks. The result was that the code protection of several μCs from different manufacturers could be bypassed with power glitching techniques, microprobing, exposure to UV-C light, or by fault injection [4]. Although this report was published in 2005, reliably protecting embedded code still appears to be an open problem, as one of our case studies shows.

In Section IV, we describe an entirely different approach to bypass the readout protection. It is possible to determine the executed instruction sequence of a CPU on the basis of the side-channel leakage. The adversary's goal is to map side-channel waveforms to the corresponding μC instructions. This can be achieved with linear [5] or nonlinear [6] pattern recognition algorithms, using principal component analysis (PCA) and Fisher's linear discriminant analysis (LDA) dimensionality reduction on the one hand, and support vector machines (SVMs) on the other hand. Both linear and nonlinear approaches lead to a recognition rate of approximately 70% on a PIC16F687, i.e., on average, 70% of the instructions of a program were recovered successfully. The recognition rate can be further improved using Markov chains or, for individual instructions, using boosting techniques for SVMs.

In Sections V and VI, we present two penetration tests against real-world applications, both equipped with standard μCs. The attack of Section V targets the digital locking and access control system SimonsVoss 3060 G2 [7]. SimonsVoss is the European market leader for electronic access control systems [8] and has sold more than 1 000 000 digital locking cylinders and 2 000 000 corresponding transponders [9]. The system 3060 is the electronic equivalent of a mechanical locking system. The term G2 identifies the latest hardware generation, which relies on a previously undisclosed, proprietary cipher and authentication protocol between transponder and digital cylinder. On both components, a Microchip PIC16F886 μC is used to implement the authentication process. Although the access to the program code for the protocol and the cryptographic algorithms was locked, the readout protection could be disabled by decapsulating the chip and exposing parts of it to UV-C light. After that, the complete program code and data of the door lock and the transponder were extracted and analyzed. Based on the reverse-engineered protocol, both cryptanalytical and side-channel attacks were developed that allow an adversary to bypass the locking system and gain unauthorized access. Section V is based on the research presented in [7] and [10].

The second security analysis (Section VI) targets Yubico's YubiKey 2 tokens [11]. They are used for a two-factor authentication, i.e., access is granted only if the user 1) knows the correct password and 2) is in physical possession of the corresponding hardware token. Typical applications are password management or computer login. Even though not a direct IoT application, it demonstrates how weaknesses of an embedded μCs can be a threat for computer network security. The YubiKey 2 provides an AES-encrypted onetime password (OTP) in addition to the standard username/password credentials. This OTP is derived on an 8-b μC manufactured by Sunplus Innovation Technology Inc. (Taiwan). As described in [12], by noninvasively measuring the electromagnetic (EM) emanation of the device, it is possible to extract the full 128-b AES key by means of SCA in less than 1 h. The attack leaves no physical traces on the device and can be performed using low-cost equipment. In consequence, an adversary is able to generate valid OTPs, even after the YubiKey 2 has been returned to the owner.

II. SIDE-CHANNEL ANALYSIS

SCA has gained considerable attention since its introduction in 1996 [1]. One reason for this is that many real-world targets were broken due to their susceptibility to side-channel attacks (e.g., [10] and [12]–[15]).

Typically, a SCA is 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. The resulting signal is often called side-channel trace or simply trace. In the second phase, the acquired traces are evaluated using digital signal processing techniques and statistical methods in order to recover a (cryptographic) secret. In Section II-A–II-C, we summarize the most common evaluation methods proposed in the literature.

A. Simple Power Analysis

In a simple power analysis (SPA) [2], one trace or an average over several traces (with identical input) to reduce the noise is usually visually inspected to directly derive a cryptographic secret. For example, let us consider a square-and-multiply algorithm to compute a Rivest–Shamir–Adleman (RSA) signature. This exponentiation algorithm consists of square operations and, depending on

the secret key, multiple multiplication operations. Generally, a squaring can be implemented more efficiently than a multiplication. This results in, e.g., a lower cycle count or a lower energy consumption. If an adversary can distinguish a multiplication from a squaring in a trace, the secret exponent could directly be read off. SPA can be applied to a variety of cryptographic algorithms, including the AES key schedule [16] and elliptic curve cryptography (ECC) scalar multiplication [17]. Additionally, SPA can be used when profiling a device under test (DUT) with an unknown implementation to prepare a key recovery. For instance, cryptographic operations often lead to an increased power consumption and can thus be identified in a trace, reducing the amount of points in time to be evaluated.

B. Template Attacks

Template attacks (TAs) are essentially an extension of SPA, replacing the manual inspection with pattern matching and machine learning techniques. In contrast to SPA, a TA requires a profiling phase, i.e., a step during which the DUT is under full control of the adversary to estimate the statistical relation between the observable random variables—in our case the respective points in time of a side-channel trace—and the internal states that are to be distinguished (for example, the value of a key byte). The resulting training set is then used to recover the desired values from a test set, i.e., traces for which the value of the key byte is considered unknown. Further details on TAs (in the context of SCA for code extraction) are given in Section IV.

C. Correlation Power Analysis

In contrast to SPA, differential power analysis (DPA) [2] and correlation power analysis (CPA) [18] are based on the evaluation of many traces with varying input data for the targeted algorithm. A brute-force attack with additional information is performed on a part of the algorithm. For instance, a DPA attack on the Data Encryption Standard (DES) or AES usually only targets the first round, for which each S-box is processed separately. Hence, each subkey entering one specific S-box can be recovered independently from the remaining key bits. By testing all possible candidates for a given subkey (e.g., $2^6 = 64$ for the DES or $2^8 = 256$ for the AES), the full key can be extracted in a divide-and-conquer manner.

For the attacks, an adversary first acquires N traces x_i, $i \in \{0, \ldots, N-1\}$, with varying input data M_i. To recover the cryptographic key, the traces are then statistically processed, e.g., using the Pearson correlation coefficient when performing a CPA.

Let K be the full key space of a block cipher. The adversary fixes a (small) subset $K_{\text{cand}} \subseteq K$ (e.g., the 256 possible 8-b subkeys entering one S-box of the AES) and considers all key candidates $k \in K_{\text{cand}}$. Then, for each $k \in K_{\text{cand}}$ and for each i, a hypothesis $V_{k,i}$ on the value of some intermediate (e.g., the output of one 8-b AES S-box) is computed. Using a power model f, this value is then mapped to $h_{k,i} = f(V_{k,i})$ to describe the process that causes the side-channel leakage. In practice, a Hamming weight (HW) or a Hamming distance (HD) power model is often suitable for complementary metal–oxide–semiconductor (CMOS) devices such as μCs [3].

In order to detect the dependency between $h_{k,i}$ and $x_i(t)$, the correlation coefficient $\rho_k(t)$ (for each point in time t and each key candidate $k \in K_{\text{cand}}$) is given as

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

with $\operatorname{var}(\cdot)$ indicating the sample variance and $\operatorname{cov}(\cdot,\cdot)$ the sample covariance according to the standard definitions [19]. 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. Considering, for example, an implementation of the AES, the attack can be performed for each S-box and the corresponding subkey separately. Instead of a complexity of $\mathcal{O}(2^{128})$ to recover the full 128-b key by an exhaustive search, this side-channel attack lowers the complexity down to $\mathcal{O}(16 \cdot 2^8)$.

D. Side-Channel Leakage of μCs compared to FPGAs and ASICs

The fundamental prerequisite for side-channel attacks is the assumption that the DUT consumes power depending on the currently processed value. For a power analysis, a basic measuring setup uses a shunt resistor in the supply path to measure the current drawn by the DUT. As an alternative, the EM emanation can be acquired by placing an EM probe close to the DUT. The physical reason for the data-dependent current consumption or EM emanation is the switching of transistors and their loads. Assuming a CMOS design, the total consumption is the sum of the static leakage current and the (for SCA more relevant) dynamic switching current. The dynamic switching current consists of the short-circuit current and the current required to charge the load capacitance of a node when the output changes from low to high. Most prominent, however, is the current caused by the activity of all attached cells, i.e., the whole combinatorial circuit switching because of a single changed bit at one input.

Usually, a μC includes a central bus connecting different distinct elements on the silicon die, e.g., to connect the computational unit to the registers, nonvolatile memory, and other peripherals. In every clock cycle, the processed intermediate value is written on the bus to be fed into the arithmetic logic unit (ALU), etc., causing current consumption to charge the bus and the load. In contrast, as ASICs or FPGAs implement the functionality directly in hardware, there is usually no requirement for a general

purpose arithmetic unit and consequent registers that have to be addressed through a bus by the memory.

In [20], different side-channel evaluation methods for the SASEBO GII (cf., [21]) built around a Xilinx Virtex-5 FPGA are compared in the form of a contest. The best attack within the contest required around 7000 traces, and later, a TA (cf. Section II-B) was presented that succeeded with only 439 traces (cf., [22]).

In contrast, for a straightforward CPA of an AES implementation on an Atmel ATMega32 μC, the full 128-b key can be recovered with approximately 50 traces—an order of magnitude lower compared to the highly optimized attack on the FPGA implementation. In general, μCs inherently exhibit a stronger side-channel leakage compared to logic implemented in hardware.

III. CODE EXTRACTION

Embedded devices are inherently exposed to product counterfeit. The main problem with protecting against this treat is that an attacker has unlimited physical access to the device and can deeply investigate its design and functionality in order to copy it. This has special relevance for devices fulfilling a security purpose like encryption or authentication: Understanding or getting to know the internal details is usually mandatory for mathematical analysis or side-channel attacks (cf., Section II).

Whenever μCs are used as central computational unit, most of the intellectual property (IP) is contained within the program code. Therefore, nearly all manufactures of μC implement methods to protect the internal memories from being readable. Those methods include single nonvolatile storage cells blocking access, physically destroyable programming interfaces (cf., [23]), or even a password protection, allowing only authorized persons to read the code (e.g., [24]).

In 2005, Skorobogatov presented a comprehensive report [4] investigating ways to circumvent or disable readout protection features. As it turned out, the protection of multiple devices of various manufactures can be bypassed using attacks with varying complexity and invasiveness. For example, some devices are vulnerable to variations of the power supply or the clock input—not requiring direct access to the silicon die. As shown in [25], even the programing interface might be logically flawed. Semi-invasive attacks directly access the silicon die, enabling optical glitches like stimulation with lasers, UV-C light, or simple photo flashes. Certainly the most sophisticated and expensive attacks belong to the fully invasive class. Here, the attackers directly alter or probe individual buses or logic cells.

In the following, we focus on products of Microchip Technology Inc. (USA), which recently delivered its 12 billionth μC [26] and is ranked second place based on the marked share of 8-b μCs according to [27]. Following the shipment figures at [28], 8-b μC can still be considered a very pervasive architecture today. Microchip divides its (flash-based) 8-b devices into the series 10F, 12F, 16F, and 18F, representing different internal architectures including varying instruction sets.

In [25], Meriac exploits a flaw in the programming interface in the PIC18FXX2/XX8 series to obtain the program code of a commercial device. These chips feature a segmented program memory with a protection bit for each segment. However, Microchip allows overwriting single segments even if the code protection is set. Consequently, it is possible to overwrite a segment with code that is able to dumb the internal code in a serial manner, circumventing the readout protection of the remaining unmodified segments. A similar program interface can be found for other models of the PIC18F series.

A semi-invasive attack on a PIC18F1320 and a PIC12F683 involving UV-C light is described in [29] and [30]. The attack is based on the method to erase today's outdated erasable programmable read only memories (EPROMs) by shining UV-C light through a small window in the package. Here, the light stimulation will erase the charge of floating gate transistors. Microchip also produced such EPROMs and, thus, knew that the configuration bits disallowing to read the program code have to be protected against UV-C light. To this end, Microchip covered the floating gate transistors with small metal plates. However, as discussed in [29], the metal plates are too small, and when shining the UV-C light at a certain angle, it will bounce off various parts, eventually hitting the floating gate.

We used this idea to test whether other models of the 10F, 12F, and 16F series are vulnerable. For example, Fig. 1(a) depicts a decapsulated PIC16F687. Following [30], the area of configuration bits as highlighted in Fig. 1(b) is depicted in Fig. 1(c). Other directly identifiable areas include the memories on the left-hand side of Fig. 1(b): the flash memory in the center, EEPROM at the top right next to the configuration bits, and static random-access memory (SRAM) in the bottom-left corner. Note that, even without using a microscope, the locations of the memories and the configuration register are distinguishable from the remaining parts of the μC (cf., Fig. 1). Thus, after identifying the relevant areas, it is possible to cover the memories in order to protect them from UV-C exposure without any magnification [cf., Fig. 1(d)]. Finally, an exposure of the silicon die to UV-C light for about 20 min resets the whole configuration register, allowing dumping the memory.

As listed in Table 1, we found that nearly all of our randomly chosen test sets of the 10F, 12F, and 16F Microchip series are vulnerable to this kind of attack. Rather, solely outdated models like the 16F84(A) were found to be resistant to UV-C light. However, Skorobogatov [4, p. 61] indicates that those μCs are vulnerable to a noninvasive voltage glitch during the erase operation. Although there is a large list of μC currently available from Microchip, the design is widely modular and only differs in size of

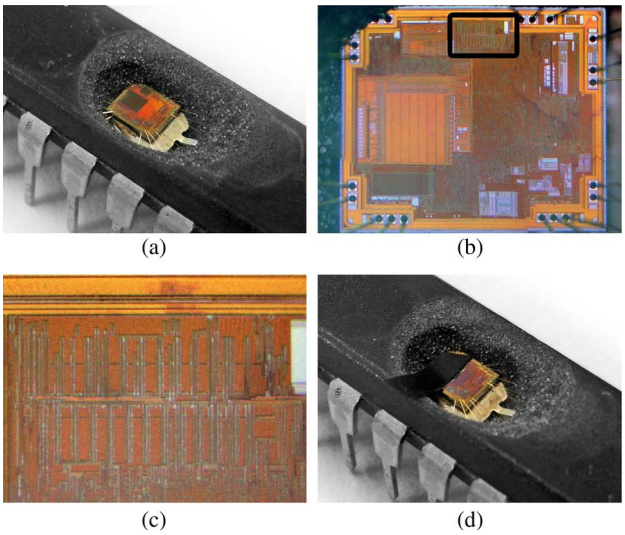


Fig. 1. *Code extraction with UV-C light of a PIC16F687: (a) μC decapsulated using white fuming nitic acid (WFNA); (b) silicon die, configuration register highlighted; (c) detail of configuration register; and (d) nonvolatile memories protected against UV-C light by isolation tape.*

memory or number of analog-to-digital converter (ADC) channels, etc. Thus, the idea of using exposure to UV-C light to reset the security mechanism is presumably applicable for most of the 10F to 16F series of Microchip. Using such a device possesses a large threat concerning product counterfeit.

IV. SIDE-CHANNEL CODE EXTRACTION

While most of the publications dealing with SCA focus on extracting the key of a cipher implementation, there exist also a few publications that concentrate on retrieving information about the executed code of a μC solely by observing its power consumption [5], [6], [31]. Interesting fields of applications are, e.g., to overcome the IP protection realized by the readout fuse bit of the μC or, from a legal point of view, the detection of product piracy. As a target device, all mentioned papers use the 8-b Microchip PIC16F687 with an instruction set of 35 instructions and an opcode size of 14 b. The core idea is to apply machine learning techniques to reconstruct the executed code, using the principles of TAs (cf., Section II-B). The code extraction problem is considered as a classical classification task which is achieved in mainly two phases.

Table 1 Microchip μC of the 10F, 12F, and 16F Series Found to Be Vulnerable to an UV-C Attack

10F200	12F519	12F629	12F683 [30]	16F73
16F88	16F627A	16F628A	16F687	16F716
16F818	16F886	16LF1826		

A. Phase I: Preprocessing

In this phase, a large number of traces are recorded for every instruction. Note that by using the term instruction, we only consider the instruction itself, e.g., ADDLW, MOVF, etc., with varying or randomized operands. This results in 41 sets of traces,[1] which is called the training set and forms the basis for the preprocessing phase. The role of preprocessing is to highlight important distinctive features that help us to distinguish between individual instructions and to discard background information or noise, which is not necessary for the classification process and would therefore only increase the computational workload. To give an example, most instructions on the PIC16F687 need four clock cycles for the execution. Depending on the instruction, within these four clock cycles, the value of the working register is read in, the computation is done, the result is stored, and the next instruction is prefetched from the instruction memory. As one can imagine, most of this information is insignificant regarding the main task of classifying the executed instruction. Thus, the goal is to focus only on relevant parts of the recordings that maximize the success with minimal effort. This can be realized by applying either linear or nonlinear feature extraction algorithms. Prominent linear feature extraction algorithms are, e.g., PCA and Fisher's LDA, and nonlinear algorithms include, e.g., kernel methods.

In a normal scenario, the data to be analyzed comprise a multitude of redundancies. In a trace, for example, the points around a peak are strongly correlated and add only little information to the amplitude of the peak. The technique of PCA tries to minimize these correlations by transforming the data into a subspace with a new basis. The axes are chosen in such a way that the first axis points to the direction of the largest variance, the second one to the largest variance under the constraint that the two axes are perpendicular, and so on. The reduction takes place by choosing only axes that are based on high variances. In [5], it is shown that considering more than 16 dimensions (compared to a few thousand dimensions before the reduction) does not improve the classification result significantly.

One disadvantage of the standard PCA is that it does not incorporate any information about the different classes. A workaround for this issue is proposed in [32] by applying a class-mean-PCA. Instead of computing the PCA of all data, it is only applied on the class means to maximize the variances between these classes. In addition to the variances between the classes, Fisher's LDA also tries to minimize the variances inside the classes. Fig. 2 shows a comparison between these three linear algorithms.

In contrast to [5], Malysiak [6] tries to solve the classification problem by using nonlinear algorithms. For

[1]Twenty five one-cycle instructions and eight two-cycle instructions, excluding the specific instructions SLEEP and RETFIE. For two-cycle instructions, every clock cycle is analyzed separately.

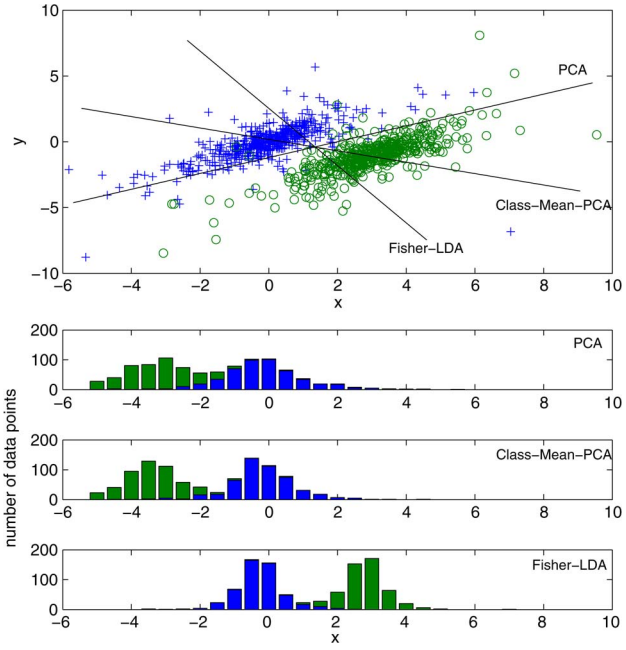


Fig. 2. *Comparison between PCA, class-mean-PCA, and Fisher's LDA. The original data set has two axes x and y. The lines in the upper plot illustrate the resulting 1-D subspaces. The overlap of the data in the subspaces is shown in the histograms below. In this example, PCA leads to a high overlap of the different classes. Better results can be achieved by considering the variances between and inside the classes [5].*

example, as a nonlinear feature extraction algorithm, the kernel PCA (KPCA) is applied instead of the linear PCA. In its core, KPCA works just like PCA. In cases where different classes cannot be separated linearly, the given data points are first transformed from the input space to a higher dimensional feature space before performing PCA computations. The idea behind this is that by choosing an appropriate transformation, the feature space allows again a linear separation of the data. This transformation is often referred to as the kernel trick.

B. Phase II: Classification Task

The second phase is the actual classification task. In this phase, newly recorded traces are assigned to specific classes using an optimal feature subset and an appropriate classifier. A simple and intuitive approach of designing a classifier is based on finding similarities between the trace of the instruction to be categorized and the training data. In [5], this is done by building templates for each instruction class composed of a class mean and a covariance matrix. A new observation is then assigned to the class with the highest posterior probability. Combined with the linear feature extraction algorithms mentioned above, this approach yields to comparable results: While the standard PCA with a dimension of 16 leads to a recognition rate of 65.2%, the class-mean-PCA results in an average rate of

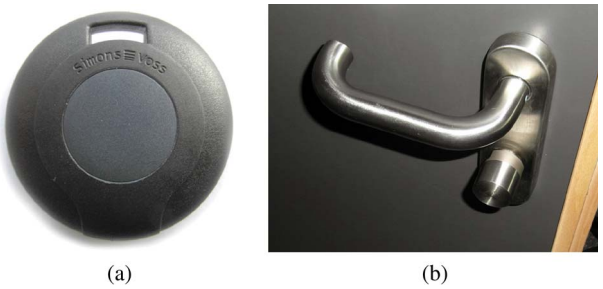


Fig. 3. *Components of the digital locking system. (a) Transponder 3064. (b) Door lock cylinder 3061.*

66.5% with 20 dimensions and Fisher's LDA in 70.1% using 17 dimensions.

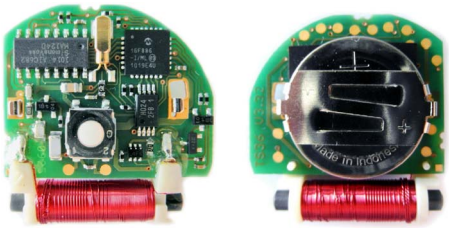
In [6], an SVM approach achieves the best results, outperforming two other classification attempts, KPCA in combination with a kernel k-nearest neighbor classification and a kernelized version of Fisher's LDA. Roughly speaking, if the classes are not linearly separable, the SVM makes use of the kernel trick. After the transformation of the data to an adequate higher dimensional feature space, a hyperplane or a set of hyperplanes are constructed that linearly separate the data with maximum distance.

The average recognition rate of the chosen SVMs is comparable to the results of the linear Fisher's LDA (70% using the same data[2]). At first glance, SVMs therefore provide no significant benefit over linear techniques which are less computational expensive. However, the SVM approach leaves much space for optimizations. In [6], a boosting technique is proposed that combines several SVMs with different recognition rates of the individual classes. For instance, while the optimal SVM with an average recognition rate of 70% achieves poor results on classifying XORWF instructions, another SVM with a lower average rate might reach better results on this instruction. The boosting technique is hence a promising approach to increase the overall recognition rate by combining the results of several SVMs.

V. CASE STUDY I: THE SIMONSVOSS DIGITAL LOCKING SYSTEM

SimonsVoss Technologies AG (Germany) is the European market leader for electronic locking and access control systems [8]. The list of customers and objects secured with this technology in Europe, the United States, and Asia, as listed on the official website [33], includes residential buildings, tourist apartments, hospitals, universities, embassies, major banks, airports, buildings of the German armed forces and the U.S. army, factory sites of well-known brands, police stations, stadiums, town halls, prisons, insurances, and many others.

[2]As denoted in [6], 64.1% refers to another data set of a different μC.



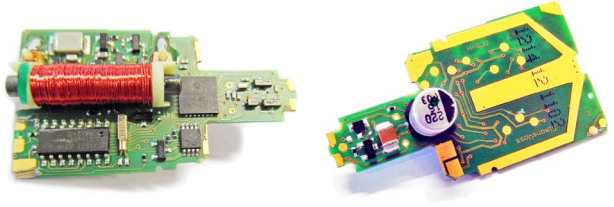
(a) (b)

Fig. 4. *PCB of transponder 3064. (a) Front. (b) Back.*

The widespread digital locking system 3060 analyzed in [7] and [10] is based on an undisclosed, proprietary cryptographic protocol. The latest revision is termed "Generation 2" or "G2-based" system by the manufacturer. It supports up to 64 000 digital locking cylinders 3061 [cf., Fig. 3(b)] per installation, up to 64 000 transponders 3064 [cf., Fig. 3(a)] per lock, and the storage of up to 1000 access instances on the transponder.

The back–end of larger installations is realized as software running on a standard personal computer (PC), allowing configuring all door locks of an installation via a wireless link, e.g., in order to reprogram the door locks. Likewise, transponders can be programmed to enable access to certain doors by means of the functionality of the back–end. In general, the "Generation 2" system enables to form a network of door cylinders, transponders, and the back–end by means of various communication techniques, e.g., through Ethernet or through multiple routers and nodes (cf., [34]). Small installations can also be configured offline, using the programming transponder 3067.

When initially analyzing the SimonsVoss system 3060, we were facing a complete black box, i.e., we had no information on the inner workings of the system. From publicly available information, little can be learned about the actual implementation. Hence, we decided to obtain the necessary knowledge for our security analysis by reverse engineering the involved components. The printed circuit boards (PCBs) of transponder and door (cf., Figs. 4 and 5) have a similar layout containing three main components that are involved in the authentication process: A SimonsVoss-proprietary ASIC is mainly responsible for modulating and demodulating the radio-frequency (RF) transmission. It is connected to a Microchip PIC16F886 μC [35], which holds the implementation of the authentication protocol. The third component is an external electrically erasable programmable read-only memory (EEPROM) controlled by the μC over an inter-integrated circuit (I^2C) bus [36].



(a) (b)

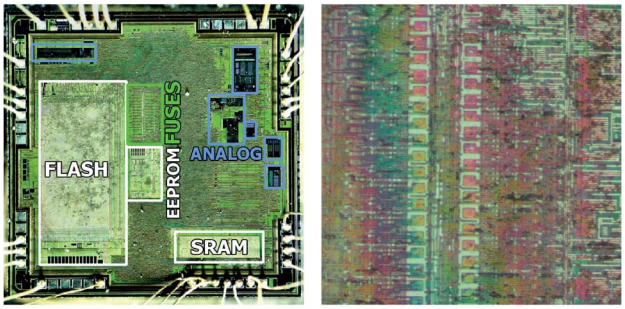
Fig. 5. *PCB of door lock 3061. (a) Front. (b) Back.*

A. Reading Out the Firmware of the PIC16F886

As described in Section III, the PIC stores its firmware, i.e., the embedded code that determines the behavior of the μC, in an internal flash memory. Moreover, the μC contains an internal EEPROM for storing 256 bytes (B) of user-defined data. In our case, after unsoldering and connecting the μC to a PicKit2 programmer [37], we read the fuses and observed that the read protection fuse for the code and internal EEPROM was enabled. Applying the methods of Section III, we tried to disable the readout protection with UV-C light.

To evaluate the vulnerability of the μC to the UV-C attack, we ordered a few blank PIC16F886 and decapsulated these test integrated circuits (ICs) with WFNA. Then, we took photographs of the silicon die of the μC with an optical microscope. An annotated image of the results is given in Fig. 6(a). To obtain an image of the area storing the fuse bits, the top layer of the silicon die was removed by sanding. The result shown in Fig. 6(b) reveals the same metal plates covering the configuration cells mentioned in [29]. However, instead of a "single line," the row appears to be cut in half with one half being rotated by 180°, forming two "mirrored lines."

We used an EPROM erasure tool as our UV-C source. After testing various positions and angles, we found that the configuration bits were erased after an exposure of about 20 min. Following Section III, the flash memory storing the program code and the internal EEPROM were protected from being deleted by covering the die with



(a) (b)

Fig. 6. *Components of decapsulated PIC16F886. (a) PIC16F886 with memories (white), analog circuits (blue), and configuration bits (green) (analog to [30]). (b) Metal shielding over floating gates (top metal layer partially removed).*

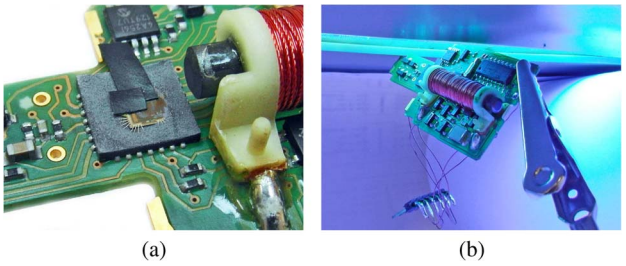


Fig. 7. *Erasing fuses of the door's PIC16F886. (a) PCB after decapsulation, area around the fuses covered with electrical isolation tape. (b) PCB exposed to the light of an UV-C EPROM erasure tool.*

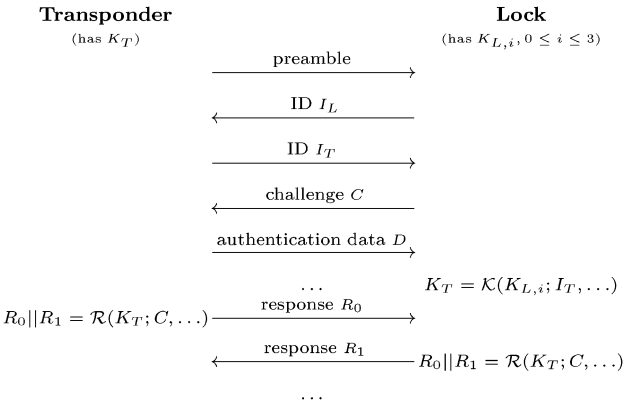


Fig. 8. *Protocol for the mutual authentication between a transponder and a lock.*

electrical isolation tape, except for the location of the configuration bits. This resulted in only the fuse bits being reset, thus disabling the code readout protection. The contents of the program memory and the EEPROM remained fully intact. This procedure was successfully applied to several PIC16F886 taken from original SimonsVoss transponders and the respective door parts. As shown in Fig. 7, it is also possible to decapsulate the PIC16F886 directly on the PCB and read it out, with the electronics remaining fully functional.

In conclusion, we were able to recover the complete firmware (stored in the internal Flash memory) and the contents of the internal EEPROM of several transponders and door lock circuits.

B. SimonsVoss Authentication Protocol

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 11 steps in total that are given in Fig. 8. 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-b key K_T is computed as the XOR of a 128-b value $K_{T,\text{int}}$ stored in the internal EEPROM of the μC (not accessible from the outside) and a 128-b 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-b 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 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 are a value sent by the transponder and control certain protocol functions. D is, however, transmitted in plaintext 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-b plaintext M under a 64-b key K.

$\mathcal{O}(Y; X)$ consists of eight rounds r, with $1 \le r \le 8$. It takes a 16-B plaintext $X = (x_0^{(1)} \ldots x_{15}^{(1)})$ and a 16-B key $Y = (y_0 \ldots y_{15})$ to compute a 16-B ciphertext $x_0^{(9)} \ldots x_{15}^{(9)}$. Fig. 9 depicts the structure of one round of $\mathcal{O}$. The internal "chaining values" $c_1^{(r)} \ldots c_{15}^{(r)}$ are processed horizontally in each round r and the last chaining value is used as the

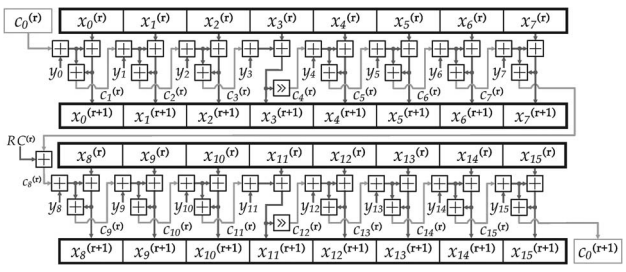


Fig. 9. *One round r of the obscurity function $\mathcal{O}$: The key bytes y_i are constant, while the $x_i^{(r)}$ and the chaining values $c_i^{(r)}$ are updated in each round. The first chaining value is $c_0^{(1)} = RC^{(0)}$. The two 8-B halves $x_0 \ldots x_7$ and $x_8 \ldots x_{15}$ are processed identically.*

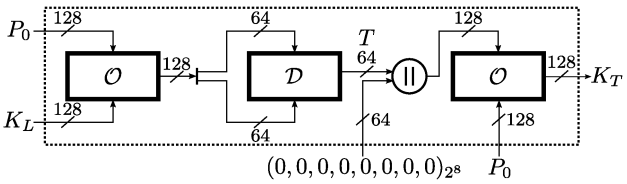


Fig. 10. *Construction of the key derivation function [7].*

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 8 B has been processed. The first chaining value is initialized with $c_0^{(1)} = RC^{(0)}$. All additions and shifts are performed modulo 256.

The key derivation function $\mathcal{K}$ takes the system key and a 128-b parameter P_0 as inputs, where P_0 is derived from the first 3 B of I_T and the first 3 B of the authentication data D as

$$P_0 = (I_{T,0}, I_{T,1}, I_{T,2}\&\texttt{0xC7}, D_0, D_1, D_2\&\texttt{0x3F}, 0, \ldots, 0).$$

The transponder key K_T is then computed by a series connection of $\mathcal{O}$, $\mathcal{D}$, and $\mathcal{O}$, as depicted in Fig. 10.

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 instance of $\mathcal{O}$ in $\mathcal{K}$ to encrypt P_0. The result is split into two 64-b halves, the lower 64 b being the plaintext and the upper 64 b being the key of $\mathcal{D}$. The 64-b 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-b response involving the challenge C and several protocol values, as shown in Fig. 11, where

$$P_1 = (C_0, \ldots, C_{10}, D_6, \ldots, D_9, 0)$$

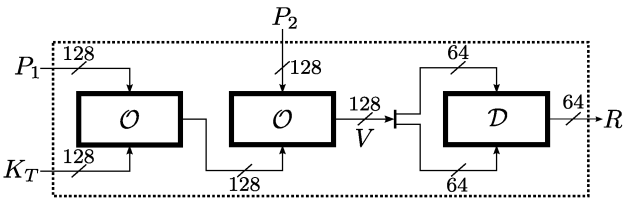


Fig. 11. *Construction of the response computation function [7].*

and

$$P_2 = (I_{L,2}, I_{T,2}, I_{T,3}, D_3, D_4, D_5, 0, \ldots).$$

The transponder sends the first 32-b 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.

An analysis of the obscurity function $\mathcal{O}$ has revealed serious weaknesses. Furthermore, it turned out that intermediate results of the response computation function were reused as part of the 11-B challenge message. Combining these two facts enables an attacker to compute a valid transponder key by only communicating with the digital cylinder. However, the manufacturer announced to fix this issue with a release of a firmware update. For more details, we refer to [7]. In the following, we focus on the susceptibility on SCA.

C. Extraction of the System Key with SCA

Obviously, the key derivation function $\mathcal{K}$ is the main target, because recovering the system key allows to derive the key of any transponder in an installation, given its I_T. In this part of the section, we describe the steps to perform a side-channel attack on 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 noninvasive, 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.

1) 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-B key $y_0 \ldots y_{15}$.

Note that only the first 6 B 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 [7]), 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 $x_i^{(r+1)} = x_i^{(r)} + c_i^{(r)} + y_i$ 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 nonrecoverable 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 $c_0^{(r)}$ up to the respective round are known, the only unknown constant in the ith update operation is y_i. The correct value can be determined using a CPA with 256 candidates for y_i in the first round, enabling to calculate the updated y_i in the targeted round.

The remaining problem with this attack is how to determine $c_0^{(r)}$. 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.

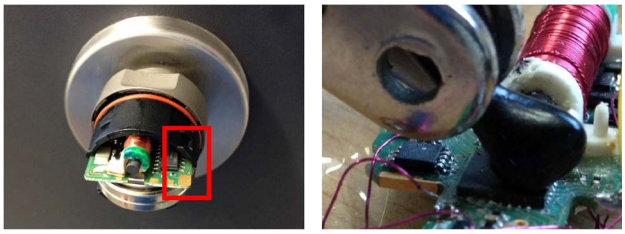


Fig. 12. ***Setup and EM trace for SCA of the door part. (a) PCB in the door part. (b) EM probe on PIC.***

2) Practical Results: In the following, we first describe the measurement setup used to acquire side-channel traces for the SimonsVoss door part 3061. By applying the attack described in Section V-C1 to real-world power traces, we recover the system key using a limited number of power traces in a noninvasive 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. 12(a)] 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. We primarily decided not to use the RF interface to increase the (mechanical) stability of the measurement setup.

We noninvasively measured the power consumption during the execution of the key derivation function for randomly chosen ID_T and authentication data 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. 12(b). 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 wideband, 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 1000 power traces using a digital storage oscilloscope at a sample rate of 500 MHz. With the current measurement setup, due to delays caused by the protocol, about ten 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

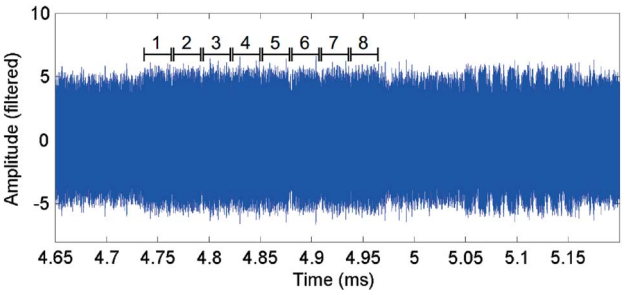


Fig. 13. *Filtered EM trace during the execution of the first obscurity function in the key derivation.*

the passband, we found that a passband from 6 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 Section V-A. This function generates a rising edge on an unused pin of the PIC, serving as a trigger signal for profiling purposes. Fig. 13 exemplary 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 Section V-C1. We use the HD between a byte $x_i^{(r)}$ and its updated value $x_i^{(r+1)}$ as the power model, i.e., $h = \mathrm{HD}(x_i^{(r)}, x_i^{(r+1)})$ (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 for the PIC series described in [31]. 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 determining the key with approximately 100 traces, as exemplarily depicted in Fig. 14(a).

The CPAs for $c_0^{(r)}$ exhibit a similar behavior; cf., for instance, Fig. 14(b) for $c_0^{(2)}$. However, for determining the partial $c_0^{(r)}$ 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

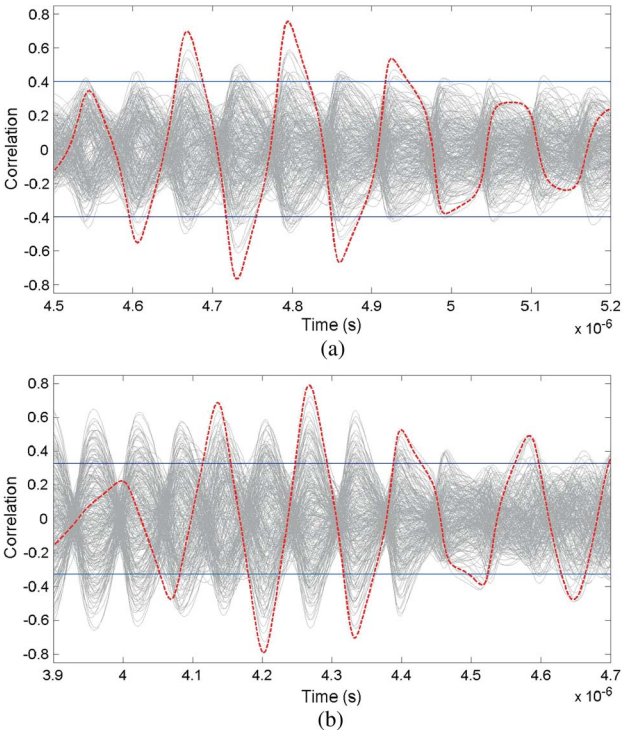


Fig. 14. *Correlation for key byte y_7 after 100 traces and $c_0^{(2)}$ after 150 traces. Correct candidate: red, dashed. (a) y_7 in round 2. (b) $c_0^{(2)}$ in round 2.*

actual key bytes in rounds 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.

VI. CASE STUDY II: THE YUBIKEY ONETIME PASSWORD TOKEN

As the second case study, we focus on a μC-based OTP token, the YubiKey 2 by Yubico [11]. In general, OTP tokens are used in scenarios where the classical way of authentication with a username/password pair is insufficient. For example, considering remote login into a corporate network, an adversary can choose from a multitude of methods to obtain the credentials, among others, by guessing passwords using a dictionary or by installing malware on the system of the target user.

The YubiKey employs an open-source protocol based on the mathematically secure AES. By emulating a USB keyboard, entering the OTP is platform independent. The token itself comprises a secret cryptographic key that, together with timestamps and counters, is used to derive a fresh OTP for each authentication. Yubico has several security-sensitive reference customers (that use the YubiKey, e.g., for securing a remote access) listed on their

website [11], e.g., Novartis, Agfa, and the U.S. Department of Defense Contractors. The U.S. Department of Defense Contractors even switched from RSA's SecureID system to the YubiKey [38], even though the YubiKey 2 is not certified for governmental standards.

A. Two-Factor Authentication

The OTP generated by the YubiKey 2 is provided in addition to the normal credentials and is only valid for a single use. Hence, the user has to know the username and password and additionally has to own the token to successfully perform an authentication. The OTP generated by the YubiKey 2 is based on several counters, random bytes, a secret ID, and a checksum, which are concatenated to a 16-B value and subsequently encrypted using the AES with a 128-b key. The exact structure of the OTP is as follows:

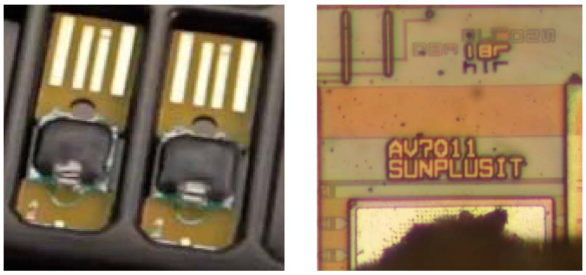
- `UID`: The private ID is 6 B long and kept secret. It can be used as another secret parameter or to distinguish users when a common encryption key is used.
- `useCtr`: The usage counter is 2 B long and increased at the first OTP generation after power-up. Additionally, the counter is incremented when the session usage counter wraps from `0xff` to `0x00`.
- `tstp`: The 3-B timestamp is initialized with random data on startup and is incremented approximately eight times per second.
- `sessionCtr`: The 1-B session counter is volatile, starts with `0x00` at power-up and is incremented on every OTP generation.
- `rnd`: Two additional bytes of random data.
- `crc`: A 2-B CRC16 checksum computed over all previous data fields.

B. Hardware of the YubiKey 2

The YubiKey 2 is mono-block molded and thus hermetically sealed. To find out which kind of μC is used in the YubiKey 2, we dissolved the casing with WFNA to gain access to the silicon die [cf., Fig. 15(a)]. The position of the μC was known from a promotional video about the production of the YubiKey [39] from which we extracted the picture of the PCB shown in Fig. 15(a). On the die, we found the label "SUNPLUSIT" [cf., Fig. 15(c)] which seems to belong to Sunplus Innovation Technology Inc. based in Taiwan [40]. We were unable to exactly find out which controller was used, as there is no Sunplus part related to the label "AV7011." However, all human interface device (HID) μCs produced by Sunplus employ an 8-b architecture.

C. Measurement Setup

To record power traces for an SCA, we built a simple adapter to get access to the USB power and data lines; cf., Fig. 16(a). The `D+` and `D−` lines are directly connected to the PC's USB port. A 60 Ω resistor was inserted into the ground line to measure the power consumption of the YubiKey 2.



(a) (b)

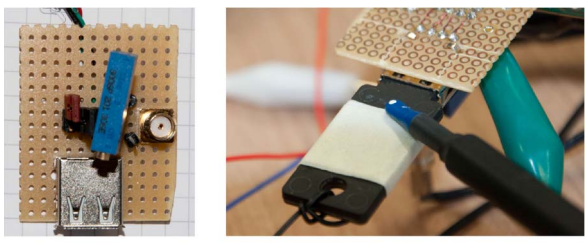
Fig. 15. *YubiKey 2 during packaging and label on the silicon die. (a) PCB (source: [39]). (b) Die label.*

In our first experiments, we used `Vcc` provided by the USB port as power supply for the YubiKey, however, this resulted in a high amount of measurement noise. Therefore, an external power supply was added to reduce the noise caused by the PC's power supply. A custom amplifier was included to amplify the measured voltage drop over the resistor. All measurements were recorded at a sample rate of 500 MHz. Fig. 17(a) depicts the overall structure of the setup.

Initially, to perform the profiling of the DUT described in Section VI-D, we focused on the power consumption measured via the shunt resistor. However, in subsequent experiments and for improving the key recovery described in Section VI-E, we also recorded the EM emanation of the DUT by placing a commercially available near-field probe [41] on the position on the package of the YubiKey 2 with the highest signal amplitude. The resulting signal was amplified by 30 dB using an amplifier made by Langer EMV [42]. The EM probe on the casing to the YubiKey 2 is depicted in Fig. 16(b).

D. Side-Channel Profiling

Initially, examining the power trace of the DUT, there was a significant voltage drop visible caused by the



(a) (b)

Fig. 16. *Measurement methods: USB adapter with shunt resistor and EM probe at the position with maximal signal amplitude on the YubiKey 2. (a) USB adapter. (b) EM probe.*

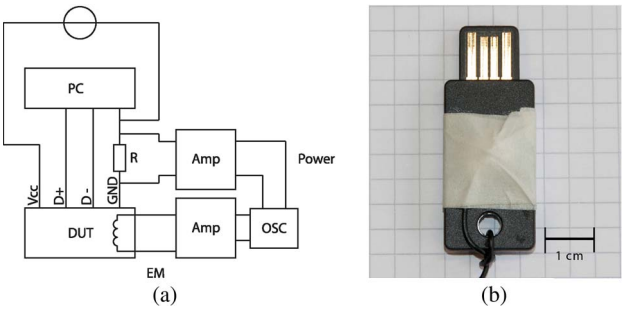


Fig. 17. *Setup for measuring the power consumption and EM emanation. (a) Schematic. (b) YubiKey 2 with wire for simulating button presses.*

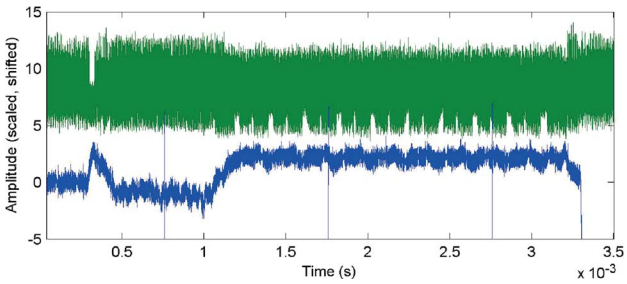


Fig. 19. *Power consumption trace (blue, bottom) and demodulated EM trace (green, top). Vertical scaling and offset changed to compare general signal shape.*

YubiKey turning off its LED. This point was used as a reference point to trigger the data acquisition of the oscilloscope. Note that at least 2.6 s are required to "recover" after a button press. Incidentally, this significantly slows down the measurement process (and thus the overall attack) because the speed of the data acquisition is limited by this property of the YubiKey.

Right before the voltage drop caused by the LED, a pattern can be observed that resembles a structure with ten rounds, each approximately 200 μs long. Since the AES-128 employed on the YubiKey has ten rounds, it is likely that this part of the trace belongs to the AES encryption. This is further confirmed by Fig. 18 showing an average trace computed using 1000 power traces. The "rounds" are clearly visible, and even different operations are distinguishable within one round. Note that, however, we were unable to observe single instructions within one round, rather, it appears the traces are in some way low-pass filtered. This might be due to a voltage regulator of the μC or decoupling capacitors. Additionally, the tenth round at approximately 2.1 ms is 70 μs shorter than the others, which agrees with the fact that the final round of the AES algorithm misses the `MixColumns` step.

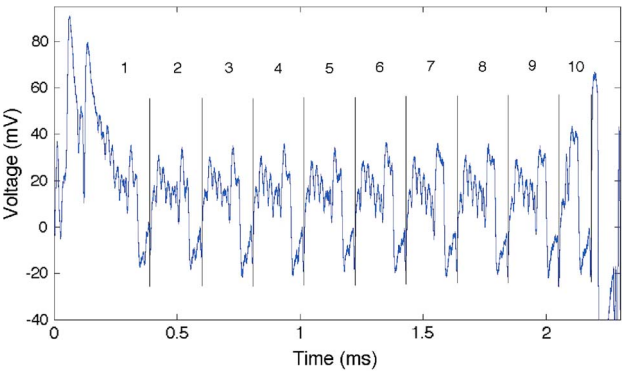


Fig. 18. *Average over 1000 amplified traces of the part suspected to belong to the AES encryption.*

We recorded 20 000 traces of the part presumably belonging to the AES operation. The 128-b AES key was set to a fixed value. The used sample rate was 500 MHz, as mentioned in Section VI-C. Experimentally, we found that (digitally) downsampling the traces by a factor of ten does not affect the success rate of the subsequent attack presented in Section VI-E. Hence, to reduce the data and computation complexity, all experiments described in the following included this preprocessing step. We tested different models for the power consumption of the device. An 8-b HW model for single bytes of the intermediate values within the AES turned out to be suitable, confirming the assumption that an 8-b μC is used in the YubiKey 2.

As mentioned in Section VI-C, we also captured the EM emanation of the DUT at the same time as the power consumption in subsequent experiments. The EM traces mainly showed a clock signal at a frequency of 12 MHz. However, digitally amplitude demodulating [43] this signal yielded a trace not exhibiting the low-pass-filtered shape observed for the power consumption traces. Fig. 19 depicts a power consumption trace (blue, bottom) and the corresponding demodulated EM trace (green, top). In both cases, the round structure is discernible. Yet, the EM trace allows to separately observe every clock cycle, while the power consumption trace only shows the overall round structure.

Similar to the power consumption traces, we also observed distorted EM traces. However, the overall number of "usable" traces was higher compared to the power consumption measurement: Only 25% of the EM traces had to be discarded, compared to about 35% of the traces acquired with the shunt.

E. Practical Attack: Extracting the AES Key

Having analyzed the round structure and identified the points in time when the leakage occurs, we continued with trying to recover the secret AES key. Initially, we used the power traces, but switched to EM traces later to reduce the number of required measurements and thus the time needed for the attack.

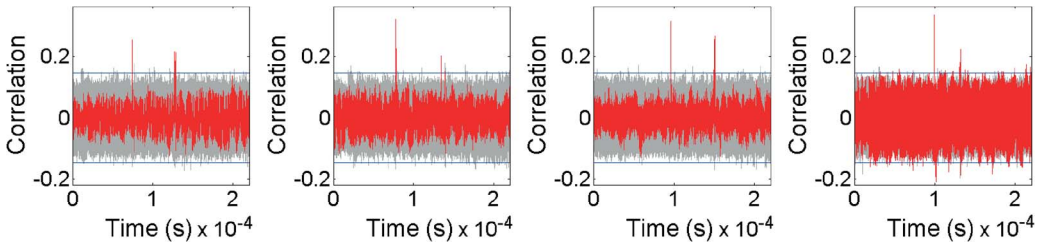


Fig. 20. *Correlation coefficient for all candidates for the key bytes 1, 2, 8, and 9 (left to right) after 800 traces. Red: correct key candidate; gray: wrong key candidates.*

We computed the correlation coefficient for all 256 candidates for each key byte using 10 000 power traces. The hypothetical power consumption h_i (cf., Section II-C) was computed as $h_i = HW(SBOX^{-1}(C_i \oplus rk))$, with C_i being a ciphertext byte (for measurement i) and rk the corresponding byte of the round key (dropping the byte index for better readability).

We were able to clearly determine the full 128-b AES key using approximately 4500 traces. In this regard, it turned out that the number of traces needed to recover a key byte differs: For byte 1, 4, and 16, less than 1000 traces were sufficient. For byte 8, 9, 10, 11, 13, and 14, less than 3000 traces sufficed to determine the correct value. For byte 2, 3, 5, 6, 7, 12, and 15, a number between 3100 and 4500 traces leads to the correct key byte being found. Note that, due to the glitches mentioned in Section VI-D, the necessary preselection of the traces effectively requires more traces to be recorded: For 4500 usable traces, approximately 7000 traces had to be measured in total. With our current measurement setup, 1000 traces can be acquired in about 1.5 h, corresponding to a rate of 11.1 traces/min. Thus, to obtain 7000 traces in total, approximately 10.5 h of access to the DUT were necessary.

We observed a "spread" correlation peak with a width of 8.3 μs, which would translate to a clock frequency of approximately 120 kHz. At this clock frequency, the execution time of about 2.5 ms (cf., Fig. 18) would imply that the AES is performed in only 300 clock cycles. Considering that even highly optimized AES implementations require about 3000 cycles on similar (and probably more powerful) 8-b μCs [44], it appears that the leakage is distributed over several clock cycles, presumably due to the low-pass characteristic mentioned in Section VI-D. Hence, we continued our analysis using the EM traces that give a higher resolution in this regard.

To this end, we performed the identical attack on the (digitally demodulated) EM traces. The resulting correlations after 800 traces for all candidates for the first, second, eighth, and ninth key bytes are exemplarily shown in Fig. 20. In contrast to the power consumption traces, the correlation for the correct key candidate clearly exceeds the one for the wrong candidate after already less than 1000 traces. Besides, the correlation peak is limited to a short instant of approximately 160 ns, which corresponds to a clock frequency of about 6.25 MHz. Thus, it is likely that this correlation is for one or a few instructions of the μC only.

To get an estimate of how many traces are needed to clearly distinguish the correct key candidate from the wrong ones, the maximum correlation coefficient (at the point of leakage) for each candidate after each trace was saved. We used the ratio between the maximum correlation for the correct key candidate and the highest correlation for the "second best" wrong candidate as a metric; cf., for instance, [45]. We then took the number of traces for which this ratio is greater than 1.1 as the minimum number of required traces. The results are given in Table 2: 500 traces are sufficient to fully recover the 128-b AES key. Due to approximately 25% of the EM traces being unusable, this translates to an overall number of around 650 traces. Thus, only 1 h of access to the YubiKey 2 is sufficient to recover the key with EM measurements, compared to 10.5 h that would be required when using the power consumption traces. Besides, a tradeoff between computation time and the number of traces could be applied. An adversary may, for instance, decide to only record 300 traces, so three key bytes (1, 3, and 14) would not be (fully) recoverable. However, these remaining three bytes, i.e., 24 b, could be easily determined using an exhaustive search on a standard PC within minutes. In this case, the measurement time is reduced to 36 min for effective 400 traces in total.

Table 2 Approximate Number of Required Traces to Recover Respective Bytes of the AES Key Using EM Traces. Metric: Ratio Between Correlation for Correct Key Candidate and Second Highest Correlation Greater Than 1.1

Key byte	1	2	3	4
# Required traces	400	300	400	200
Key byte	5	6	7	8
# Required traces	300	200	300	200
Key byte	9	10	11	12
# Required traces	200	200	200	200
Key byte	13	14	15	16
# Required traces	300	500	300	300

Given the AES key, an adversary is able to generate an arbitrary number of valid OTPs and thus to impersonate the legitimate owner given that the traditional credentials have been obtained, e.g., by means of eavesdropping, phishing, or malware. Note that a standard CPA was sufficient to mount our attack with a number of traces small enough to pose a threat in the real world. Hence, we did not further investigate more complicated (profiled) SCA techniques like TAs, as described in Section II.

Further, the here presented attack leaves no physical traces on the DUT. The only means by which the attack could be detected is a (relatively high) increase of the usage counters (cf., Section VI-A). Due to the fact that the volatile session counter has to reach 256 first before the nonvolatile usage counter is incremented, the EM-based attack only increases the usage counter by two when recording 500 traces. Thus, the presented attack does not lead to a "suspicious" change of this counter and is very unlikely to be detected in this way.

VII. CONCLUSION

A. Summary

In this paper, we demonstrated that standard μCs are vulnerable to numerous attacks, including the extraction of the embedded code (Sections III and IV) and of cryptographic secrets by means of SCA (Section II). Consequently, using off-the-shelf μCs for applications with moderate to high security requirements may render the resulting system insecure, even if the employed cryptographic primitives (e.g., AES) are secure from a mathematical point of view.

The case studies of Sections V and VI give examples in this regard: In both cases, implementation attacks could be used to extract cryptographic keys, giving an adversary access to secured buildings (Section V) or, e.g., protected web services and networks (Section VI), respectively. Note that the susceptibility of real-world devices toward SCA and related techniques is a general problem—further examples for vulnerable devices include KeeLoq door openers [13], secure EEPROMs made by Atmel [46], Mifare DESFire MF3ICD40 contactless smartcards [15], and FPGAs by Xilinx, Actel, and Altera [47]–[50].

Nevertheless, in terms of security, standard μCs mark the lower end: While the attacks presented in this paper work with less than 150 (SimonsVoss lock) and 700 traces (YubiKey 2), respectively, SCA of more secure (hardware) implementations may require several tens or hundred thousand traces to succeed [15], [50]. Moreover, reverse-engineering vendor-specific functionality by reading out embedded code is usually easier (in terms of required time, knowledge, and equipment) than analyzing FPGA configuration bitstreams or directly the silicon die of ICs.

Finally, we would like to stress that SimonsVoss and Yubico had been notified well in advance of the disclosure of the respective weaknesses. To our knowledge, both companies attempt to improve the security of their products.

B. Countermeasures and Future Work

As a major point for future work, the question arises how to protect embedded (cryptographic) systems against the presented attack methods. With respect to SCA, various countermeasures have been proposed, with an overview given in [3]: As a first step, one may attempt to eliminate the leakage on the level of the circuit itself. To this end, custom logic styles to balance the power consumption have been designed; cf., for example, [51]. However, in practice, it turns out that completely removing the side-channel leakage is nearly impossible, e.g., due to manufacturing tolerances and so on. Thus, SCA-based attacks still apply, but may require a higher number of traces. The same holds for countermeasures based on generating additional noise, e.g., by randomly switching a large load in parallel to the execution of the cryptographic operation. In general, balancing the power consumption reduces the signal-to-noise ratio (SNR) (decreasing the leakage signal), while the generation of noise achieves the same by increasing the amplitude of the noise.

In addition to focusing on the amplitude dimension, the time dimension can also be used to impede SCA. Varying the clock signal or inserting "dummy" operations leads to a desynchronization of the traces, increasing SNR and thus increasing the amount of traces needed for an attack. On the algorithmic level, the order of certain (independent) operations (e.g., S-Box lookups in a block cipher) may be randomized, a technique referred to as shuffling in the literature.

Finally, the intermediate values within the state of the algorithm itself can be randomized by adding a so-called "mask" that varies for each execution of the algorithm. Such masking schemes [52], [53] remove the dependency between a predicted intermediate and the observed side-channel signal altogether, however, often come with significant (time and/or space) overhead and thus can be too costly for resource-constrained devices. Thus, although numerous countermeasures against implementation attacks and SCA in particular have been proposed in the literature, further investigations in this area are of interest, especially when it comes to protecting low-cost and thus significantly constrained devices such as μCs.

With respect to the extraction of the embedded code of μCs, both offensive and defensive aspects deserve further attention. Most countermeasures available today are designed to prevent highly invasive attacks: For example, (active) meshes cover the silicon die and the data bus itself is encrypted to impede microprobing. Nevertheless, even high-security devices protected with such countermeasures can still be attacked, as, for example, demonstrated for the Infineon SLE 66PE that was successfully attacked by Tarnovsky [54] using a focused ion beam (FIB).

Moreover, new methods like side-channel code extraction (cf., Section IV) or the use of the photonic side channel [55] (which gives a spatially resolved map of the activity of an IC) have likely not been considered in the development process of secure ICs so far. Devising at the same time effective and cost-efficient countermeasures against these threats is hence a highly relevant area for future research. ■

REFERENCES

[1] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, other systems," *CRYPTO 1996*, vol. 1109, N. Koblitz, Ed. Berlin, Germany: Springer-Verlag, 1996, pp. 104–113, ser. Lecture Notes in Computer Science.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *CRYPTO 1999*, vol. 1666, M. J. Wiener, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 388–397, ser. Lecture Notes in Computer Science.

[3] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer-Verlag, 2007.

[4] S. P. Skorobogatov, "Semi-invasive attacks—A new approach to hardware security analysis," Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-630. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf

[5] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a side channel based disassembler," *Trans. Comput. Sci.*, vol. 10, pp. 78–99, 2010.

[6] D. Malysiak, "GPU assisted implementation of kernel-based template attacks," M.S. thesis, Horst Görtz Institute for IT-Security, Ruhr Univ. Bochum, Bochum, Germany, 2011. [Online]. Available: http://www.emsec.rub.de/media/crypto/attachments/files/2011/08/Thesis_Malysiak_Darius.pdf

[7] D. Strobel *et al.*, "Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system," *CRYPTO 2013*, vol. 8042, R. Canetti and J. A. Garay, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 147–164, ser. Lecture Notes in Computer Science.

[8] Industrytoday, "SimonsVoss leads European access control market," 2010. [Online]. Available: http://www.industrytoday.co.uk/security/simonsvoss-leads-european-access-control-market/2227

[9] SimonsVoss Technologies AG, "The finest in keyless security," 2013. [Online]. Available: http://www.simons-voss.de/fileadmin/media/Imagebroschueren/SimonsVoss_I mage_EN.pdf

[10] D. Oswald, D. Strobel, F. Schellenberg, T. Kasper, and C. Paar, "When reverse-engineering meets side-channel analysis—Digital lockpicking in practice," *Selected Areas in Cryptography*. Berlin, Germany: Springer-Verlag, 2013, pp. 571–588.

[11] Yubico, "Two-factor authentication with the YubiKey," 2013. [Online]. Available: http://www.yubico.com

[12] D. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the Yubikey 2 one-time password generator," *Research in Attacks, Intrusions, Defenses*, vol. 8145, S. J. Stolfo, A. Stavrou, and C. V. Wright, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 204–222, ser. Lecture Notes in Computer Science.

[13] T. Eisenbarth *et al.*, "On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme," *CRYPTO 2008*, vol. 5157. Berlin, Germany: Springer-Verlag, 2008, pp. 203–220, ser. Lecture Notes in Computer Science.

[14] T. Kasper, M. Silbermann, and C. Paar, "All you can eat or breaking a real-world contactless payment system," *Financial Cryptography 2010*, vol. 6052. Berlin, Germany: Springer-Verlag, 2010, pp. 343–350, ser. Lecture Notes in Computer Science.

[15] D. Oswald and C. Paar, "Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world," *CHES 2011*, vol. 6917. Berlin, Germany: Springer-Verlag, 2011, pp. 207–222, ser. Lecture Notes in Computer Science.

[16] S. Mangard, "A simple power-analysis (SPA) attack on implementations of the AES key expansion," *ICISC 2002*, vol. 2587. Berlin, Germany: Springer-Verlag, 2002, pp. 343–358, ser. Lecture Notes in Computer Science.

[17] C. D. Walter, "Simple power analysis of unified code for ECC double and add," *CHES 2004*, vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 191–204, ser. Lecture Notes in Computer Science.

[18] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *CHES 2004*, vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 16–29, ser. Lecture Notes in Computer Science.

[19] E. W. Weisstein, "Variance," Mathworld—A Wolfram Web Resource. Dec. 2010 [Online]. Available: http://mathworld.wolfram.com/Variance.html

[20] J.-L. Danger *et al.*, "DPA contest v2: Rules of the contest." [Online] Available: http://www.dpacontest.org/v2/rules.php

[21] Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, "Side-channel attack standard evaluation board SASEBO-GII specification," 2009. [Online]. Available: http://www.dpacontest.org/v2/hall_of_fame.php

[22] DPA Contest v2, "Hall of Fame." [Online]. Available: http://www.dpacontest.org/v2/hall_of_fame.php

[23] Texas Instruments Inc., "MSP430 Programming Via the JTAG Interface—Users Guide," 2013. [Online]. Available: http://www.ti.com/lit/ug/slau320k/slau320k.pdf

[24] Infineon Technologies AG, "TC1797 32-bit Single-Chip Microcontroller Data Sheet V1.2 2009-09," 2009.

[25] M. Meriac, "Heart of darkness—Exploring the uncharted backwaters of HID iCLASS security," in *Proc. 27th Chaos Commun. Congr.*, 2010. [Online]. Available: http://www.openpcd.org/HID_iClass_demystified

[26] Microchip Technology Inc., "Microchip technology delivers 12 billionth PIC microcontroller to leading motor manufacturer," Nidec Corporation, 2013. [Online]. Available: http://www.microchip.com/pagehandler/en-us/press-release/microchips-12-billionth-pic-mi.html

[27] Microchip Technology Inc., "Investor Presentation," 2013. [Online]. Available: http://www.microchip.com/investor/Pressrelease/MCHP tation.060313.pdf

[28] IC INSIGHTS, INC., "MCU market on migration path to 32-bit and ARM-based devices," 2013. [Online]. Available: http://www.icinsights.com/data/articles/documents/541.pdf

[29] A. Huang, "Hacking the PIC 18F1320," 2005. [Online]. Available: http://www.bunniestudios.com/blog/?page_id=40

[30] A. Zonenberg, "Microchip PIC12F683 Teardown," 2011. [Online]. Available: http://siliconexposed.blogspot.de/2011/03/microchip-pic12f683-teardown.html

[31] M. Goldack, "Side-channel based reverse engineering for microcontrollers," Diploma thesis, Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum, Germany, 2008. [Online]. Available: https://www.emsec.rub.de/media/attachments/files/2012/10/da_goldack.pdf

[32] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," *CHES 2008*, vol. 5154, E. Oswald and P. Rohatgi, Eds. Berlin, Germany: Springer-Verlag, 2008, pp. 411–425, ser. Lecture Notes in Computer Science.

[33] SimonsVoss Technologies AG, "References," 2012. [Online]. Available: http://www.simons-voss.com/References.1163.0.html?&L=1

[34] SimonsVoss Technologies AG, "Manual for WAVENET-FUNKNETZWERK 3065," 2011. [Online]. Available: http://www.simons-voss.de/fileadmin/media/produkte/Handbuch_WaveNet_Fun knetzwerk_3065_D.pdf

[35] Microchip Technology Inc., "PIC16F882/883/884/886/887 Data Sheet," 2009. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/41291f.pdf

[36] NXP Semiconductors, "User manual UM10204—I^2C-bus specification and user manual, rev. 5," 2012. [Online]. Available: http://www.nxp.com/documents/user_manual/UM10204.pdf

[37] Microchip Technology Inc., "PICkit 2 development programmer/debugger." [Online]. Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&5nodeId=1406&dDocName=en023805

[38] Yubico, "Yubico reference customers: Department of Defense." [Online]. Available: http://www.yubico.com/about/reference-customers/department-defence/

[39] Yubico, "How YubiKeys are manufactured." [Online]. Available: https://www.youtube.com/watch?v=s8_I1-ErZSQ

[40] Sunplus Innovation Technology Inc. [Online]. Available: http://www.sunplusit.com

[41] Langer EMV-Technik, "LF1 near field probe set." [Online]. Available: http://www.langer-emv.de/en/products/disturbance-emission/near-field-probes/lf-1/

[42] Langer EMV-Technik, "Preamplifier PA 303." [Online]. Available: http://www.langer-emv.de/en/products/disturbance-emission/preamplifier/pa-203-303/devices-data/

[43] K. S. Shanmugam, *Digital & Analog Communication Systems*. Mumbai, India: Wiley-India, 2006.

[44] J. W. Bos, D. A. Osvik, and D. Stefan, "Fast implementations of AES on various platforms," IACR Cryptology ePrint Archive, 2009, p. 501. [Online]. Available: http://eprint.iacr.org/2009/501

[45] D. Oswald and C. Paar, "Improving side-channel analysis with optimal linear transforms," *Smart Card Research and Advanced Applications*, vol. 7771, S. Mangard, Ed. Berlin, Germany: Springer-Verlag, 2013, pp. 219–233, ser. Lecture Notes in Computer Science.

[46] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede, "Power analysis of Atmel CryptoMemory—Recovering keys from secure EEPROMs," *CT-RSA 2012*, vol. 7178, O. Dunkelman, Ed. Berlin, Germany: Springer-Verlag, 2012, pp. 19–34, ser. Lecture Notes in Computer Science.

[47] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs," in *Proc. ACM Conf. Comput. Commun. Security*, 2011, pp. 111–124.

[48] A. Moradi, M. Kasper, and C. Paar, "Black-box side-channel attacks highlight the importance of countermeasures—An analysis of the Xilinx Virtex-4 and Virtex-5 bitstream encryption mechanism," *CT-RSA 2012*, vol. 7178. Berlin, Germany: Springer-Verlag, 2012, , pp. 1–18, ser. Lecture Notes in Computer Science.

[49] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," *CHES 2012*, vol. 7428, E. Prouff and P. Schaumont, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 23–40, ser. Lecture Notes in Computer Science.

[50] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 2013, pp. 91–100.

[51] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proc. 28th Eur. Solid-State Circuits Conf.*, 2002, pp. 403–406.

[52] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *CRYPTO 1999*, vol. 1666, M. Wiener, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 398–412, ser. Lecture Notes in Computer Science. [Online]. Available: http://dx.doi.org/10.1007/3-540-48405-1_26

[53] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against differential power analysis," *CHES 2000*, vol. 1965, C. K. Koç and C. Paar, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 231–237, ser. Lecture Notes in Computer Science.

[54] C. Tarnovsky, "Deconstructing a 'secure' processor," Presentation at Black Hat USA, 2010. [Online]. Available: http://www.blackhat.com/html/bh-dc-10/bh-dc-10-speaker_bios.html#Tarnovsky

[55] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," *CHES 2012*, vol. 7428, E. Prouff and P. Schaumont, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 41–57, ser. Lecture Notes in Computer Science.

ABOUT THE AUTHORS

Daehyun Strobel received the B.S. degree in applied computer science and the M.S. degree in IT security from Ruhr University Bochum, Bochum, Germany, in 2006 and 2009, respectively, where he is currently working toward the Ph.D. degree with the Chair for Embedded Security.

His research interests include practical side-channel attacks on embedded systems and side-channel reverse engineering of embedded software.

David Oswald received the Ph.D. degree in IT security from the Horst Görtz Institute for IT-Security at the Ruhr University Bochum, Bochum, Germany, in 2013.

He is currently working for the Chair for Embedded Security, Ruhr University Bochum, Bochum, Germany. His main field of research is the practical security analysis of embedded systems. His focus is on attack methods that exploit weaknesses in the physical implementation of mathematically secure cryptographic algorithms. Those techniques include both (passive) side-channel analysis and (active) fault injection. He is the cofounder of Kasper & Oswald GmbH, Bochum, Germany, offering innovative products and services for security engineering.

Bastian Richter received the B.S. degree in IT security from Ruhr University Bochum, Bochum, Germany, in 2013, where he is currently working toward the M.S. degree in IT security.

He is a student assistant under the Chair for Embedded Security. His research interests include security of embedded systems with focus on side-channel analysis and fault injection.

Falk Schellenberg received the B.S. and M.S. degrees in IT security from Ruhr University Bochum, Bochum, Germany, in 2010 and 2012, respectively, where he is currently working toward the Ph.D. degree in physical security for the Chair for Embedded Security, under the supervision of C. Paar.

His research interests include (semi)invasive attacks, especially (optical) fault injection techniques, as well as practical attacks on real-world devices.

Christof Paar (Fellow, IEEE) received the M.Sc. degree from the University of Siegen and the Ph.D. degree from the Institute for Experimental Mathematics at the University of Essen, Germany.

He holds the Chair for Embedded Security at Ruhr University Bochum, Bochum, Germany, and is an Affiliated Professor at the University of Massachusetts Amherst, Amherst, MA, USA. He cofounded, with C. Koc, the Workshop on Cryptographic Hardware and Embedded Systems (CHES) series. He has over 150 peer-reviewed publications and is coauthor of the textbook *Understanding Cryptography* (New York, NY, USA: Springer-Verlag, 2010). He is a cofounder of ESCRYPT—Embedded Security, a leading consultancy firm in applied security that is now part of Bosch. His research interests include highly efficient software and hardware realizations of cryptography, physical security, security evaluation of real-world systems, and cryptanalytical hardware.