Side-Channel Analysis of Keymill

Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel

Graz University of Technology, Austria
`christoph.dobraunig@iaik.tugraz.at`

Abstract. One prominent countermeasure against side-channel attacks, especially differential power analysis (DPA), is fresh re-keying. In such schemes, the so-called re-keying function takes the burden of protecting a cryptographic primitive against DPA. To ensure the security of the scheme against side-channel analysis, the re-keying function has to withstand both simple power analysis (SPA) and differential power analysis (DPA). Recently, at SAC 2016, Taha et al. proposed Keymill, a side-channel resilient key generator (or re-keying function), which is claimed to be inherently secure against side-channel attacks. In this work, however, we present a DPA attack on Keymill, which is based on the dynamic power consumption of a digital circuit that is tied to the $0 \rightarrow 1$ and $1 \rightarrow 0$ switches of its logical gates. Hence, the power consumption of the shift-registers used in Keymill depends on the $0 \rightarrow 1$ and $1 \rightarrow 0$ switches of its internal state. This information is sufficient to obtain the internal differential pattern (up to a small number of bits, which have to be brute-forced) of the 4 shift-registers of Keymill after the nonce has been absorbed. This leads to a practical key-recovery attack on Keymill.

Keywords: side-channel analysis · fresh re-keying · differential power analysis

1 Introduction

Side-channel attacks like differential power analysis (DPA) pose a serious threat to devices operating in a hostile environment. Such scenarios quite naturally appear in our current information infrastructure whenever an entity has physical access to a device which uses a cryptographic key that must be kept secret from this entity. Hence, it is necessary to protect such devices against the extraction of the secret key by means of side-channel analysis like SPA and DPA [7]. In particular, for resource-constrained or low-cost devices that are used for the Internet of Things or in RFID applications, the use of protection mechanisms is not straightforward, since applied protection mechanisms have to be cheap and efficient. One protection mechanism that suits such applications very well is fresh re-keying.

Fresh re-keying [9] is an approach for precluding DPA on cryptographic primitives. The resistance against DPA is achieved by a separation-of-duties principle, where a re-keying function takes the burden of protection against DPA away from the cryptographic primitive. In this construction, the re-keying function

This article is a minor revision of an article that appears in the proceedings of COSADE 2017. The final publication is available at Springer via `http://dx.doi.org/10.1007/978-3-319-64647-3_9`.

processes a nonce and master key to compute a fresh session key. This session key is then used by the cryptographic primitive. The nonce, or initial value (IV), is generated uniquely for each encryption, and must never be reused for another encryption. The nonce is considered public information and has to be transmitted to (or synchronized with) the decrypting recipient together with the ciphertext. Since the cryptographic primitive is only called once per session key, DPA attacks are naturally prevented, and only dedicated countermeasures against SPA are needed. However, the re-keying function has to provide resistance against SPA and DPA attacks, either by its design, or by application of countermeasures like threshold implementations [10], masking [12], hiding [2], shuffling [6], etc. The intention behind re-keying schemes is that the re-keying function itself can be protected more easily against DPA than the cryptographic scheme, or that it can even be designed to provide inherent security against DPA. Both options profit from the fact that the re-keying function itself does not need to fulfill strong cryptographic requirements [9].

Re-keying functions. Medwed et al. [9] proposed polynomial multiplication as re-keying function, which has further been extended to the multi-user setting [8]. While such a polynomial multiplication lacks inherent protection against DPA, it is easy to mask and additionally allows easy-to-implement countermeasures against SPA, such as shuffling [9]. However, Pessl and Mangard [11] showed at CT-RSA 2016 that this multiplication is vulnerable to side-channel analysis, in particular at the point where its masks have to be combined and the session key is used in the cryptographic scheme. Additionally, the original scheme by Medwed et al. is susceptible to time-memory trade-off attacks [3]. Recently at Crypto 2016, Dziembowski et al. [4] presented a more formal treatment of re-keying functions and proposed two schemes. The first is based on learning parity with leakage, the second on learning with rounding, and both are efficient and easy to mask.

Keymill. In contrast to designs relying on side-channel countermeasures like masking for side-channel protection, Keymill [14] claims to be secure against side-channel analysis inherently by design without requiring any redundant circuit. Having a re-keying function which provides inherent security against side-channel analysis is beneficial with respect to implementation metrics. Since such schemes do not require masking to withstand DPA, no randomness is needed to create and update masks, and masks do not have to be stored and processed in the first place. A comparison of a modular multiplication and Keymill by Taha et al. [14] shows that a hardware implementation of Keymill requires 775 gate equivalents (GE), while an implementation of a modular multiplication with first-order masking requires 7300 GE [9].

To achieve such low implementation costs, Keymill only uses 4 nonlinear feedback shift-registers taken from the stream cipher Achterbahn [5]. The shift-registers are connected via a rotating cross-connect, which shifts the output of each shift-register's nonlinear feedback function into another shift-register. This cross-connect joins the function outputs with shift-register inputs cyclically per clock. For this construction and also for a toy example consisting of two 8-bit registers involving a similar rotating cross-connect, the authors claim that no DPA attacks are feasible without constructing a hypothesis for the whole key, or equivalently for the whole internal state of the four shift-registers, and thus render DPA attacks infeasible.

Our Contribution. In this work, we present a DPA attack on Keymill. Our attack shows that the claim of Keymill to be inherently secure against side-channel attacks without the need of additional circuits does not hold. The basic idea of the attack is as follows. Instead of making a hypothesis about the exact values of the internal state bits or the secret key, we target the internal difference between neighboring bits of the shift-registers. As observed by Burman et al. [1], and Zadeh and Heys [15], the dynamic power consumption of shift-registers depends on the number of internal differences of neighboring bits. The more internal differences we have, the more power the shift-register consumes. We recover those internal differences bit by bit by comparing the power consumption of a reference nonce (e.g., 0), with power traces of a modified nonce where a single bit has been flipped. Knowing these internal differences allows to recover the full state and consequently the master key by guessing a few additional bits.

Our attack requires the attacker to obtain traces for related (partially chosen) pairs of nonce values, but without violating the single-use requirement for nonces. This scenario is explicitly covered by the security claim of Keymill, although similar to chosen-plaintext attacks, it might not be easy to collect such data in a practical application. We verified the validity and robustness of the attack both for simulated data and for measurements from an FPGA implementation of Keymill.

Outline. In Sect. 2, we give a brief background on fresh re-keying and restate the specification of Keymill. Then, we describe the side-channel attack on Keymill and on a variant of Toy Model II given in the Keymill specification in Sect. 3. Sect. 4 gives experiments for our attack and discusses the influence of different levels of noise. Finally, we conclude in Sect. 5.

2 Background

In this section, we first give a brief introduction to the concept of fresh re-keying, where we restate the requirements on re-keying functions. Then, we briefly summarize the specification of Keymill and finally, discuss time-memory trade-off attacks on such re-keying schemes.

2.1 Fresh Re-Keying

Fresh re-keying has been proposed by Medwed et al. [9] as a countermeasure against side-channel and fault attacks for low-cost devices. A typical scenario where fresh re-keying can be applied is the communication of an RFID tag with an RFID reader. Typically, RFID tags are low-cost devices that additionally have strict requirements regarding power consumption, not allowing costly protection mechanisms against side-channel and fault attacks of the implemented cryptographic primitives. This stands in contrast to the more expensive RFID readers, where costly protection mechanisms like masking are usually affordable.

Fig. 1 shows the working principle of fresh re-keying in a communication scenario between an RFID reader and an RFID tag. For sending a message, the tag generates a nonce and derives a session key k^* by using a re-keying function g. This session key is then used by the block cipher E to encrypt the message m. The ciphertext c together with the nonce is sent to the reader, where it can be decrypted.

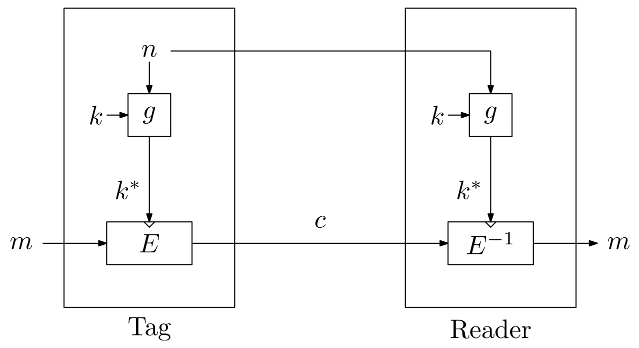

Fig. 1. Fresh re-keying scheme of Medwed et al. [9].

Since the nonce is generated by the tag, the tag can ensure that the block cipher E is always used with a new session key k^*, which will preclude DPA on the block cipher. However, in the case of the reader, having a unique nonce cannot be ensured, because the nonce is received over the communication channel and thus, might be chosen by an attacker. Therefore, the implementation of the block cipher E of the reader has to be protected against DPA by other means. Apart from that, the implementation of g for both entities has to withstand DPA, because here, the master key k is processed with a different nonce. On the designer's side, the challenge is to find a suitable re-keying function g which fulfills the following six properties given by Medwed et al. [9]:

1. Good diffusion of the master key k.
2. No synchronization between parties. Hence, g should be stateless.
3. No need for additional key material.
4. Little hardware overhead. Total costs lower than protecting E alone.
5. Easy protection against side-channel attacks.
6. Regularity.

One option for a re-keying function is the polynomial multiplication in $\mathbb{F}_{2^8}[y]$ modulo $p(y)$ proposed by Medwed et al. [9]:

$$g \,:\, (\mathbb{F}_{2^8}[y]/p(y))^2 \to \mathbb{F}_{2^8}[y]/p(y), \quad (k,n) \mapsto k \cdot n.$$

2.2 Brief Description of Keymill

Keymill [14] is a new keystream generator recently proposed by Taha et al. at SAC 2016. In contrast to the fresh re-keying scheme by Medwed et al. discussed in Sect. 2.1, Keymill does not only provide one session key k^*, instead it provides a keystream. As indicated in Fig. 2, this is particularly useful when encrypting longer messages that require several block cipher calls. The nonce n is required to be unique, but is otherwise public.

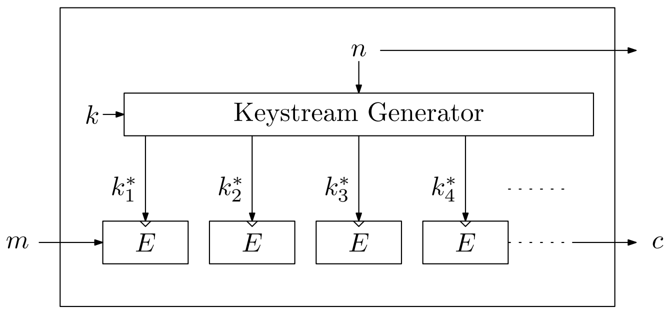

Fig. 2. Re-keying using a keystream generator as shown in [14].

Keymill operates on an internal state of 128 bits, composed of 4 NLFSRs as shown in Fig. 3. Shift-register R_0 has 31 bits, shift-registers R_1 and R_2 have 32 bits, and shift-register R_3 has 33 bits. The feedback functions F_0, F_1, F_2 and F_3 are selected from the set of feedback functions used for the stream cipher Achterbahn [5]:

$$\begin{aligned}
F_0(S) &= s_0 + s_2 + s_5 + s_6 + s_{15} + s_{17} + s_{18} + s_{20} + s_{25} + s_8 s_{18} + s_8 s_{20} \\
&+ s_{12} s_{21} + s_{14} s_{19} + s_{17} s_{21} + s_{20} s_{22} + s_4 s_{12} s_{22} + s_4 s_{19} s_{22} \\
&+ s_7 s_{20} s_{21} + s_8 s_{18} s_{22} + s_8 s_{20} s_{22} + s_{12} s_{19} s_{22} + s_{20} s_{21} s_{22} \\
&+ s_4 s_7 s_{12} s_{21} + s_4 s_7 s_{19} s_{21} + s_4 s_{12} s_{21} s_{22} + s_4 s_{19} s_{21} s_{22} \\
&+ s_7 s_8 s_{18} s_{21} + s_7 s_8 s_{20} s_{21} + s_7 s_{12} s_{19} s_{21} + s_8 s_{18} s_{21} s_{22} \\
&+ s_8 s_{20} s_{21} s_{22} + s_{12} s_{19} s_{21} s_{22} \\
F_1(S) = F_2(S) &= s_0 + s_3 + s_{17} + s_{22} + s_{28} + s_2 s_{13} + s_5 s_{19} + s_7 s_{19} \\
&+ s_8 s_{12} + s_8 s_{13} + s_{13} s_{15} + s_2 s_{12} s_{13} + s_7 s_8 s_{12} + s_7 s_8 s_{14} \\
&+ s_8 s_{12} s_{13} + s_2 s_7 s_{12} s_{13} + s_2 s_7 s_{13} s_{14} + s_4 s_{11} s_{12} s_{24} \\
&+ s_7 s_8 s_{12} s_{13} + s_7 s_8 s_{13} s_{14} + s_4 s_7 s_{11} s_{12} s_{24} + s_4 s_7 s_{11} s_{14} s_{24}
\end{aligned}$$

$$\begin{aligned}F_3(S) = {} & s_0 + s_2 + s_7 + s_9 + s_{10} + s_{15} + s_{23} + s_{25} + s_{30} + s_8 s_{15} + s_{12} s_{16} \\ & + s_{13} s_{15} + s_{13} s_{25} + s_1 s_8 s_{14} + s_1 s_8 s_{18} + s_8 s_{12} s_{16} + s_8 s_{14} s_{18} \\ & + s_8 s_{15} s_{16} + s_8 s_{15} s_{17} + s_{15} s_{17} s_{24} + s_1 s_8 s_{14} s_{17} + s_1 s_8 s_{17} s_{18} \\ & + s_1 s_{14} s_{17} s_{24} + s_1 s_{17} s_{18} s_{24} + s_8 s_{12} s_{16} s_{17} + s_8 s_{14} s_{17} s_{18} \\ & + s_8 s_{15} s_{16} s_{17} + s_{12} s_{16} s_{17} s_{24} + s_{14} s_{17} s_{18} s_{24} + s_{15} s_{16} s_{17} s_{24}\end{aligned}$$

Note that all feedback functions are nonsingular and additionally do not depend on the first bit $s_{\ell-1}$ of each ℓ-bit register, that is, they are of the form

$$F_j(S) = F_j(s_0, \ldots, s_{\ell-1}) = s_0 + F'_j(s_1, \ldots, s_{\ell-2}).$$

The outputs of the feedback functions are then mixed via a rotating cross-connect, depending on the current clock cycle index i:

$$F_j \to R_{j+i \pmod 4} \quad \text{for } j = 0, 1, 2, 3.$$

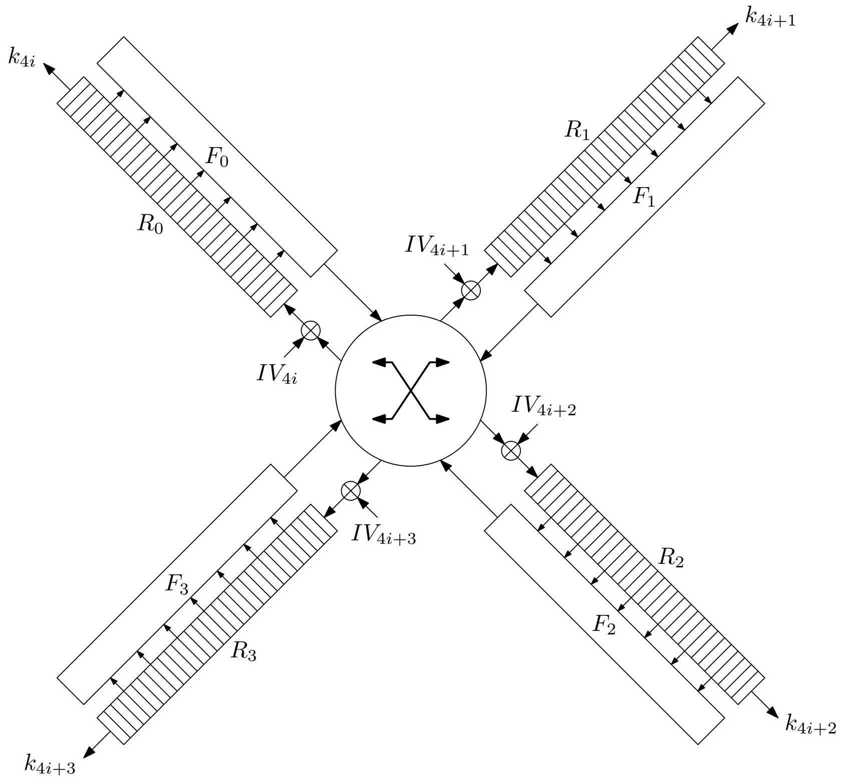

Fig. 3. Structure of Keymill

After loading the 128-bit secret key into the internal state, 4 bits of the 128-bit nonce that can be monitored (or controlled) by the attacker are added to the feedback functions of the shift-registers in each clock cycle. After absorbing the nonce in 32 clock cycles, the internal state is clocked 33 more times before producing any output. Afterwards 4 bits of output are generated (one from each shift-register) in each clock cycle. We refer to the specification of Keymill [14] for a more detailed description.

The designers claim that this construction "expands the size of any useful key hypothesis to the full entropy" [14]. More specifically, they claim that the SCA-security ("the minimum size of a key hypothesis (in bits) such that the leakage-model using the correct key correlates to the measured leakage significantly higher than the leakage-model using any other key" [14]) is about 128 bits.

2.3 Remark on Time-Memory Trade-Off Attacks

As elaborated in [3], the re-keying scheme proposed by Medwed et al. [9] is susceptible to time-memory trade-off attacks dependent on the used re-keying function. For instance, if a polynomial multiplication is used together with AES-128, the master key can be recovered with a complexity of 2^{65} [3]. Since Keymill has an internal state-size of 128-bits, similar attacks are possible on the scheme shown in Fig. 2.

3 Side-Channel Attack on Keymill

In this section, we will present side-channel attacks on Keymill. First, we discuss the power consumption of shift-registers following the work of Zadeh and Heys [15] and show how this power consumption can be used to recover the differences of neighboring shift-register bits. This and the fact that the first bits of the shift-registers are not used in the feedback functions of Keymill allows us to mount a side-channel attack. For simplicity, we first demonstrate the attack on a variant of Toy Model II given in the Keymill specification [14] and afterwards discuss the application to Keymill.

3.1 Power Consumption of a Shift-Register

In all our attacks, we exploit the dynamic power consumption of the shift-registers at the triggering edge of the clock (i.e., positive edge). More specifically, we observe the dynamic power consumption of the building blocks of the shift-registers, the D-flip-flops. As shown by Zadeh and Heys [15], the dynamic power consumption of a D-flip-flop at the triggering edge depends on whether its state changes or not. If the state of the D-flip-flop changes, more power is consumed than if it remains the same. As an example, Zadeh and Heys [15] analyze a D-flip-flop constructed out of 6 NAND gates. For such a flip-flop, 3 gates change if the flip-flop changes its state, whereas only one gate changes if not.

Next, we have a look at the power consumption of a shift-register. For simplicity, consider a 4-bit shift-register consisting of 4 flip-flops D_0, D_1, D_2, and D_3. In the following, we assume that D_4 is the input of our shift-register, which is shifted towards D_0. For instance, let us consider the power consumption of the change from state $S_0 = \texttt{0110}_2$ to state $S_1 = \texttt{1101}_2$. For this transition, D_0 changes its state, D_1 keeps its state, D_2 changes its state, and D_3 changes its state. Since the power consumption of the flip-flops is higher if they change their state, the power consumption of the shift-register is correlated with the Hamming weight of $S_0 \oplus S_1$ $(= \texttt{1011}_2)$. In this example, 3 flip-flops change their state.

Now, we want to consider a state change from S_0 to S'_1, where we shift in a 0 instead of a 1 as before. So we observe the power consumption for the change from state $S_0 = \texttt{0110}_2$ to state $S'_1 = \texttt{1100}_2$. If this transition happens, only two flip-flops change their state. Thus, we observe for the transition $S_0 \to S'_1$ a smaller power consumption than for $S_0 \to S_1$. This allows us to derive information about the difference of the bits stored in D_4 and D_3 of S'_1 and S_1, respectively. In more detail, we know that they are equal for S'_1 and different for S_1. We will use this observation in our side-channel attack on a variant of Toy Model II and Keymill itself in the following sections.

3.2 Attack on Toy Model II

For the sake of simplicity, we first describe the working principle of our attack on a slightly modified variant of Toy Model II given in the Keymill specification [14], which has only two 8-bit shift-registers. In the attack, we assume that similar to Keymill, the output of the first flip-flop of each shift-register is not connected to the feedback function, as shown in Fig. 4. Besides nonsingularity, this is the only assumption on the feedback function that is necessary to mount our attack. We do not rely on any other specific properties of the feedback functions. The shift-register is preinitialized with the secret key. After that, the 16-bit nonce is absorbed, 2 bits per clock cycle. Our goal is to recover all *internal differences* of both shift-registers after the nonce (e.g., $n = \texttt{0000}_{16}$) has been absorbed.

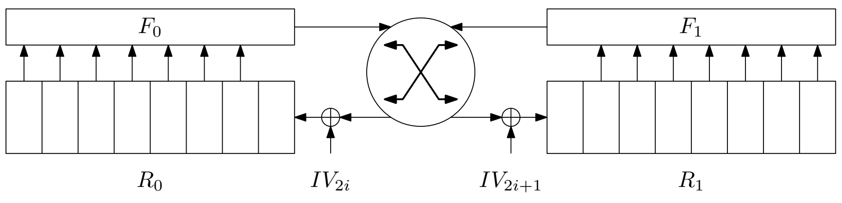

Fig. 4. Structure of modified Toy Model II

First, we collect two power traces, one for a nonce starting with $\texttt{00}_2$ and one for a nonce starting with $\texttt{10}_2$. We look at the power consumption when the first two bits of the nonce are absorbed in the first cycle. Here, we have a difference in n_0 for R_0, but equal values in n_1 for R_1. Since the first flip-flop of each shift-register is not connected to the feedback function, the circuit processes the same information for both initial values, except for the first flip-flop of the left shift-register R_0. As already discussed in Sect. 3.1, this gives us information about the difference of the first two bits of R_0 after absorbing the first two bits of the nonce. If the power consumption when absorbing $\texttt{00}_2$ is higher than in the $\texttt{10}_2$ case, we know that the first two bits of R_0 are different after $\texttt{00}_2$ is absorbed. If the power consumption is lower, then they are equal.

Next, we use two initial values starting with $\texttt{00}_2$ and $\texttt{01}_2$. This allows us to learn the internal difference of the first two bits of the shift-register R_1 after $\texttt{00}_2$ is absorbed. Then, we use $\texttt{0000}_2$ and $\texttt{0010}_2$ to learn information of the difference of the first two bits after $\texttt{0000}_2$ has been absorbed, still preserving the information of the difference of the now second and third bits of both shift-registers learned in the steps before. By continuing in this way, we can learn the differences of all neighboring bits of R_0 and R_1 after the nonce $\texttt{0000}_{16}$ has been absorbed.

Now, guessing one bit in each shift-register determines the other 7 bits in each shift-register. Hence, we are left with only 4 possible internal states. From this states on, we can invert Toy Model II step by step until we get 4 key candidates in total. Note that inversion of a fully known state is trivial due to the nonsingularity of the feedback functions, which allows to recover the previous last bit s_0 from the known feedback output and the known values of the other taps. Overall, if we are able to obtain noiseless measurements for about 16 chosen nonces (one per bit of the state), we can recover the entire key k.

3.3 Attack on Keymill

Compared to Toy Model II, Keymill is essentially the same, except everything is larger. As described in Sect. 2.2, we have 4 shift-registers: one 31-bit shift-register, two 32-bit shift-registers, and one 33-bit shift-register. The 128-bit nonce is absorbed in 32 cycles, each cycle taking 4 bits. Furthermore, the 4 feedback functions of Keymill do not consider the outputs of the first flip-flop of each shift-register. As mentioned before, this fact is exploited in our side-channel attack. Again, we want to recover the internal differential pattern of the used shift-registers after a certain nonce, e.g., $n = 0 \cdots 0$ has been absorbed. Please note that the all 0 nonce is just an example taken for simplicity. The attack works for every other choice of the nonce.

The attack proceeds in a similar way as described in Sect. 3.2. First, we record a power trace for a nonce starting with $\texttt{0000}_2$ and a second trace for a nonce starting with $\texttt{1000}_2$. We compare the power consumption for the two traces at the time the first nibble of the nonce is absorbed. At this time, for both traces, the processed values are equal except for the inputs of shift-register R_0. Since the output of the first flip-flop of R_0 is not fed back into the feedback function, the power consumption differs only because of the state changes of this flip-flop. As discussed in Sect. 3.1, this is sufficient to recover the difference of the first two bits of shift-register R_0. The power traces of nonces starting with $\texttt{0100}_2$, $\texttt{0010}_2$, and $\texttt{0001}_2$ can be used to learn the difference of shift-registers R_1, R_2 and R_3, respectively.

When the second nibble of the nonce is absorbed, those differences are shifted by one position, but are still known, if the first nibble of the nonce starts with $\mathtt{0000}_2$. Hence, we can use nonces starting with $\mathtt{0000\,0000}_2$, $\mathtt{0000\,1000}_2$, $\mathtt{0000\,0100}_2$, $\mathtt{0000\,0010}_2$, and $\mathtt{0000\,0001}_2$ and learn the differences of the first two bits of each shift-register, while retaining the knowledge of the differences between the second and the third bits. Proceeding this way, we can learn at most 32 differences of neighboring bits per shift-register.

This means that we can learn all internal differences of all 4 shift-registers, since one shift-register has 31 bits, two have 32 bits and one has 33 bits. So, at most 30, two times 31, and 32 differences have to be learned. Since we know all internal differences of each shift-register, a guess of one state bit in each shift-register determines all others. Thus, guessing 4 bits in total leads to 16 different states we recover. From these states, we can invert Keymill, resulting in 16 possible key candidates in total.

Summarizing, if we can obtain noiseless measurements for about 128 chosen nonces, then we can recover the full internal state and consequently the secret key k. In particular, we recover the internal state bit by bit by making a hypothesis on 1 bit of "equivalent key information", instead of an actual key bit value: The xor difference of two neighboring state bits.

3.4 A Note on Filtering the Noise

The success of our attacks crucially depends on the ability to distinguish power consumption changes for a change of the input values. This means that the noise level has to be small enough to reliably identify these changes. If the attacker is allowed to repeat nonces, averaging the traces and filtering the noise is no problem. Even if the nonce is required to be unique (as usually the case), this can easily be done, since the state of the shift-registers only depends on bits of the nonce that have already been absorbed. Hence, we can use all the remaining nonce bits after the relation we want to recover to average the power consumption for this cycle. For Keymill, we can average over up to 16 power traces even if we recover bit relations in the penultimate nonce-absorbing cycle. Dependent on the noise level, it might happen that the last few internal differences of the state cannot be recovered anymore, since there are too few traces to filter the noise. So these bits might have to be guessed additionally at the end of the attack.

4 Practical Evaluation

In order to show the practicability of the attacks discussed in Sect. 3, we present two experiments. First, we run the attack based on simulated leakage traces to analyze the impact of noise on the success of the attack. For the second evaluation, we use power measurements from an FPGA implementation of Keymill to evaluate the practicability of the attack targeting real hardware.

First, we simulate the described attack targeting the proposed Keymill design as shown in Fig. 3. Therefore, the four registers $R_0 \ldots R_3$ and the corresponding feedback functions $F_0 \dots F_3$, which compose the four NLFSRs, have been modelled in software. At the start of the simulation, the registers are initialized with the secret key. Then, for every clock cycle, the simulation returns the Hamming distance produced by the shift registers. The current Hamming distance depends on the values in the shift register, the results of the feedback functions $F_0 \dots F_3$ and the nonce.

Gaussian noise with zero mean ($\mu_{\text{noise}} = 0$) and varying standard deviation σ_{noise} can be added to the noise-free Hamming-distance measurements ($\text{HD}_{\text{noisefree}}$) in order to simulate measurements captured from real hardware, i.e. HD_{meas} (see Equation 1). In order to minimize the influence of the noise it is possible to repeat the simulation with a similar nonce t times for calculating the mean of the measurements.

$$\text{HD}_{\text{meas}} = \text{HD}_{\text{noisefree}} + \text{noise}, \qquad \text{where noise} \leftarrow \mathcal{N}(0, \sigma_{\text{noise}}). \tag{1}$$

For every setting (specific σ_{noise} and specific t), we performed $N_{\text{full}} = 500$ experiments with randomly chosen initial states of the four shift-registers $R_0 \dots R_3$ to calculate the success rate SR of the attack,

$$\text{SR} = \frac{N_{\text{success}}}{N_{\text{full}}},$$

where N_{success} is the number of successful state recoveries. Fig. 5 depicts the results of this simulation. It is clearly visible that SR decreases with increasing noise. This effect can be compensated by repeating the attack with the same nonce t times and calculate the mean of the measurements. For $t = 1$, the success rate starts to decrease for noise levels above $\sigma_{\text{noise}} = 0.1$. For $t = 50$, the success rate remains 1 up to a noise level of $\sigma_{\text{noise}} = 1.3$.

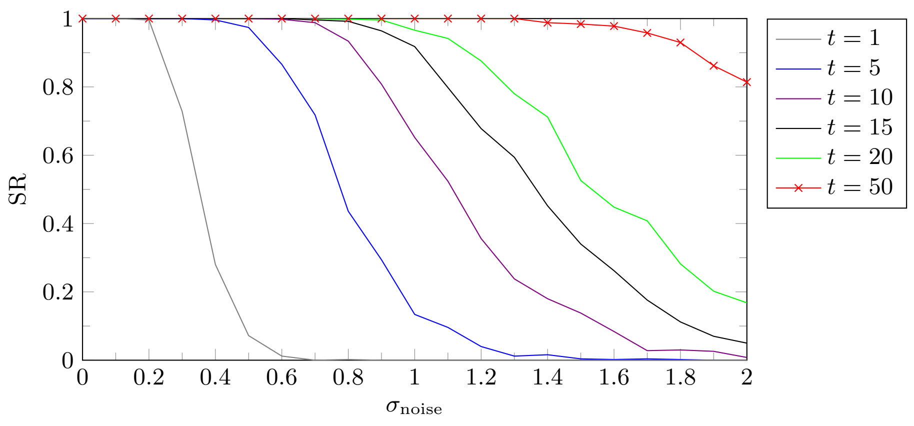

Fig. 5. Success rate (SR) for increasing noise levels (σ_{noise}). For the graphs different numbers (1–50) of Hamming-distance measurements have been used for calculating the mean Hamming distance.

Fig. 6 shows the influence of σ_{noise} on the Hamming-distance measurements (HD_{meas}). For this specific plot, $\text{HD}_{\text{noisefree}} = 64$ has been selected. The '+' markers represent single HD measurements. In the noise-free scenario, i.e. $\sigma_{\text{noise}} = 0$, all HD measurements have the value 64. For a high noise level, i.e. $\sigma_{\text{noise}} = 2$, the HD measurements are in the range between 58 and 70.

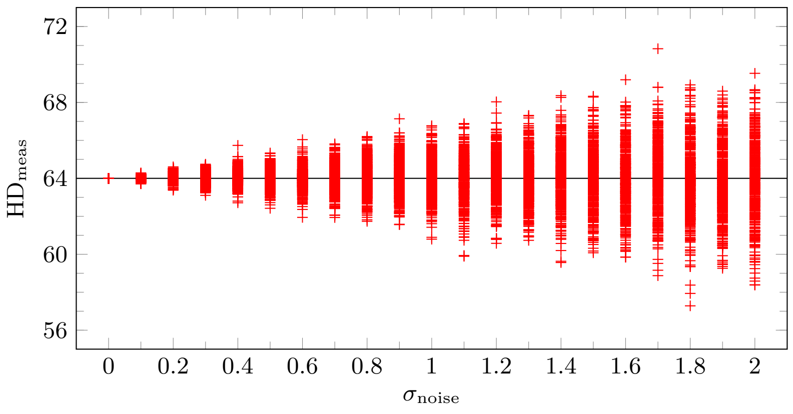

Fig. 6. Hamming distance measurements for increasing σ_{noise}, $\text{HD}_{\text{noisefree}} = 64$.

In a final experiment, Keymill is evaluated on real hardware. We chose the Sakura G board [13], which is the reference platform for side-channel evaluations of cryptographic hardware designs on FPGAs. The main FPGA (Xilinx Spartan-6 LX75) has been configured with the Keymill design and the power consumption during the initialization (i.e. the first 33 clock cycles where the bits of the nonce are shifted into the shift registers, four bits per clock cycle) has been measured with an oscilloscope. For every bit position of the nonce, two trace sets have been recorded, one with the corresponding bit set to '`0`' and one with the corresponding bit set to '`1`'. In order to evaluate the number of traces required for reaching a specific success rate, 10 000 traces have been recorded for every nonce. The results of the evaluations are depicted in Fig. 7. It shows that for the given FPGA implementation, at least 220 measurements for every nonce are required for reaching a success rate of 1. In scenarios where repeated measurements of the same nonce are prohibited, iterating over the last 8 bits of the nonce can be done to average the measurements. This leads to 256 traces per fixed 120 bits that can be used to filter the noise.

Comparing the means of the two trace sets allows to distinguish between the Hamming distances. The higher amount of traces required for reaching a success rate of 1 indicates that the noise on real hardware is significantly larger than the noise during previously performed simulations. For the sake of completeness we have performed the simulations for $t = 220$ and larger noise levels. The results show that for $\sigma_{\text{noise}} \geq 3.6$ the success rate starts to decrease for $t =$

220. Experiments on the real hardware reveal that for recovering the whole initial state, approximately $220 \cdot 128 = 28\,160$ measurements are required in total. The applied measurement setup allows us to collect the required amount of measurements for reaching a success rate of 1 within an hour. With some improvements of the setup the measurement time could be reduced to a few minutes, but this was not the goal of this work.

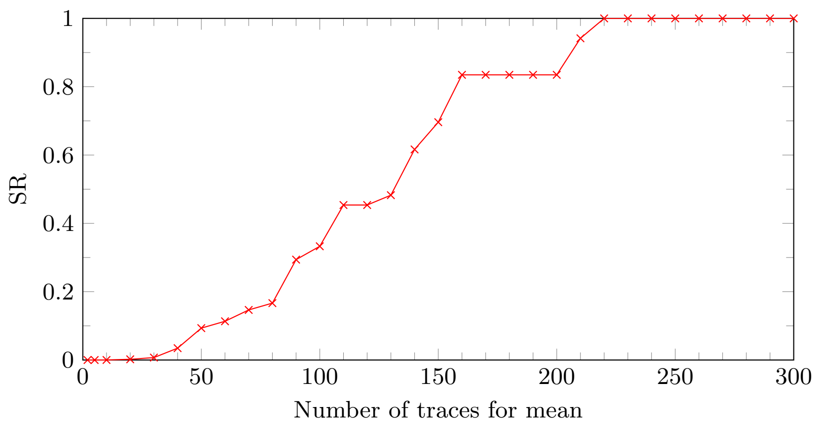

Fig. 7. Evolution of the success rate (SR) for the attack on the FPGA with increasing number of traces for calculating the mean.

5 Conclusion

In this work, we showed that a DPA attack on Keymill is feasible. In contrast to the DPA attacks that are claimed to be thwarted by the specification of Keymill, we do not make hypotheses on the actual values of Keymill's key or internal state. Instead, we first recover the internal differences of neighboring bits step by step from side-channel measurements, and then take advantage of the resulting entropy reduction to recover the actual values. Our attack violates the claim by the designers that Keymill is inherently secure against side-channel attacks by design. Indeed, we show that Keymill needs dedicated countermeasures against DPA attacks exploiting internal differences.

Our attack requires the ability of an attacker to choose the nonces. Therefore, guaranteeing that only random nonces can be used seems to be an efficient countermeasure. Although this prevents a straightforward application of our attack to recover all differences between state-bits, the recovery of just a fraction of the differences of the first few bits still remains possible. Hence, it is part of future work to evaluate if extensions of the presented attack concept are applicable for random nonces.

Acknowledgments. This work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS).

References

1. Burman, S., Mukhopadhyay, D., Veezhinathan, K.: LFSR based stream ciphers are vulnerable to power attacks. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 384–392. Springer (2007)
2. Clavier, C., Coron, J.S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer (2000)
3. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F.: On the security of fresh re-keying to counteract side-channel and fault attacks. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 233–244. Springer (2014)
4. Dziembowski, S., Faust, S., Herold, G., Journault, A., Masny, D., Standaert, F.X.: Towards sound fresh re-keying with hard (physical) learning problems. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 272–301. Springer (2016)
5. Gammel, B.M., Göttfert, R., Kniffler, O.: Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project (2006)
6. Herbst, C., Oswald, E., Mangard, S.: An AES smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252 (2006)
7. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 388–397. Springer (1999)
8. Medwed, M., Petit, C., Regazzoni, F., Renauld, M., Standaert, F.X.: Fresh re-keying II: Securing multiple parties against side-channel and fault attacks. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 115–132. Springer (2011)
9. Medwed, M., Standaert, F.X., Großschädl, J., Regazzoni, F.: Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer (2010)
10. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of non-linear functions in the presence of glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer (2008)
11. Pessl, P., Mangard, S.: Enhancing side-channel analysis of binary-field multiplication with bit reliability. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 255–270. Springer (2016)
12. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer (2013)
13. Sakura-G – Side-Channel Evaluation Board. `http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html`, accessed: 2016-11-28
14. Taha, M., Reyhani-Masoleh, A., Schaumont, P.: Keymill: Side-channel resilient key generator. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, Springer (2016), (to appear). eprint version: `http://eprint.iacr.org/2016/710`
15. Zadeh, A.A., Heys, H.M.: Simple power analysis applied to nonlinear feedback shift registers. IET Information Security 8(3), 188–198 (2014)