Protecting PUF Error Correction by Codeword Masking

Dominik Merli[1], Frederic Stumpf[1], Georg Sigl[2]

[1] Fraunhofer Research Institution for Applied and Integrated Security (AISEC)
Munich, Germany
{dominik.merli,frederic.stumpf}@aisec.fraunhofer.de

[2] Institute for Security in Information Technology
Technische Universität München, Munich, Germany
sigl@tum.de

Abstract. One of the main applications of Physical Unclonable Functions (PUFs) is unique key generation. While the advantages of PUF-based key extraction and embedding have been shown in several papers, physical attacks on it have gained only little interest until now. In this work, we demonstrate the feasibility of a differential power analysis attack on the error correction module of a secure sketch. This attack can also be applied to code-offset fuzzy extractors because they build upon secure sketches. We propose a codeword masking scheme to protect key generation algorithms used for PUFs. Our proposed countermeasure enables masking of linear Error-Correcting Codes (ECCs) without impact on their error correction capabilities while keeping the overhead low. This is achieved by random masking codewords, which can be efficiently generated by the ECC's encoding function. Further, it allows to consistently protect the PUF-based key generation process and can provide the masked key and its mask to a subsequent crypto module which implements masking as well. We demonstrate the practical protection of our codeword masking scheme by attacking a masked secure sketch implementation. We emphasize that, besides protecting code-offset algorithms, the proposed masking scheme can also be applied to index-based syndrome coding and other security-critical error correction modules.

Keywords: Physical Unclonable Functions, Error-Correcting Codes, Masking, Secure Sketch, Fuzzy Extractor, Differential Power Analysis

1 Introduction

Physical Unclonable Functions (PUFs) exploit unique information generated by sub-micron manufacturing variations of mainly, but not only, silicon devices. They have increasingly gained interest in academia as well as industry since they provide promising security features for low-cost devices, such as Radio Frequency Identification (RFID) tags [5], but also for high-security products like smartcards [7].

Until now, key generation as a basis for conventional cryptographic algorithms is the dominating practical application. There, a cryptographic key is bound to or generated from noisy measurements of physical properties like ring oscillators' frequencies [24] or SRAM cells' start-up values [9].

One of the most popular key generation algorithms is the *code-offset fuzzy extractor* proposed by Dodis et al. [6], which is based on *code-offset secure sketches* [6]. Secure sketches as well as fuzzy extractors usually utilize linear Error-Correcting Codes (ECCs) to achieve reliability, because raw PUF measurements naturally involve a certain amount of noise. During an enrollment phase, they generate helper consisting of a code-offset between the secret PUF bits and an ECC codeword. This helper data is later used for reliable reconstruction of the initial key. Since this reconstruction phase is performed, when devices are out in the field, physical attacks, e.g. power analysis [15,19], have to be considered for the security of PUF-based devices.

In this contribution, we show that code-offset secure sketches, the basic building block of code-offset fuzzy extractors, can be attacked by correlation-based DPA because of leakage originating from the error correction module. Applying standard masking is not possible for ECCs because, thereby, they would loose their error correction capabilities. Therefore, we propose a codeword masking scheme to still be able to protect linear ECCs against DPA and show how to apply it to code-offset constructions. The effectiveness of the proposed masking scheme is evaluated by a correlation-based DPA attack on the error correction of a protected secure sketch design on an FPGA. Further, we generalize the threat of DPA based on helper data manipulations to other PUF key generation algorithms and give an overview of the range of applications where our codeword masking scheme can improve protection.

The paper is organized as follows. In Section 2, we give an overview of related work. Section 3 shows the exploitation of ECC side-channel leakage of a secure sketch implementation. In Section 4, we propose codeword masking, show how to apply it to code-offset architectures and evaluate its effectiveness. Section 5 give a broader view on where codeword masking can be applied, followed by a conclusion in Section 6.

2 Related Work

There exists some related work about ECCs causing vulnerabilities in security critical systems. In 2009, Dai and Wang [4] presented a study on side-channels of ECCs used in reliability enhancing techniques for

memories. Also, side-channel attacks on the McEliece public key cryptosystem [23,21] exploit leakage of ECC implementations. We show that side-channel leakage of ECCs in PUF-based key generation algorithms have to be considered as realistic threat.

Until now, only a few publications deal with side-channel analysis of secure sketches and fuzzy extractors. Karakoyunlu et al. [13] presented two attacks on software implementations of Reed-Solomon codes [17] and Bose-Chaudhuri-Hocquenghem (BCH) codes [17]. The authors claimed that standard software implementations of these codes show data dependent leakage, which can be attacked by SPA. Further, they proposed a differential template attack, where they built templates for every possible input symbol by varying the helper data. Then, a set of distinguished templates is chosen for the analysis of the device under attack. The work of Schuster [22] and Merli et al. [20] showed that the Toeplitz hashing [16], which is part of previously proposed efficient fuzzy extractor implementations, can be broken by SPA. They also theoretically explain the possibility to exploit side-channel information generated by the code-offset XOR upon helper data manipulations. We practically demonstrate the feasibility of DPA attacks on code-offset constructions and, in contrast to earlier attack papers, propose a countermeasure to protect secure sketches and fuzzy extractors against SPA and first-order DPA attacks.

3 Attacking the Error Correction of a Secure Sketch

Since its introduction in 1996 [14], side-channel analysis of cryptosystems has been an important topic for embedded security. One of the most powerful tools to attack software and hardware implementations of cryptographic algorithms is DPA [15,19]. In this section, we demonstrate the applicability of correlation-based DPA to a code-offset secure sketch. The shown vulnerability of the ECC implementation also holds for code-offset fuzzy extractors because they are only extensions of secure sketches.

3.1 Code-Offset Secure Sketch

One of the basic key generation algorithms is a secure sketch [6] based on a code-offset construction. In the following, we use it to embed and reconstruct a secret key into random PUF reponse bits.

During an enrollment phase, an external key $\mathbf{k}$ is encoded to a codeword $\mathbf{c}$ of a code C: $\mathbf{c} = \mathit{encode}_C(\mathbf{k})$. Then, the public helper data (the sketch) $\mathbf{w} = \mathbf{c} \oplus \mathbf{r}$ is calculated as the code-offset between $\mathbf{c}$ and the initial

PUF response vector $\mathbf{r}$. During the reconstruction phase, the *Rec* module, as shown in Figure 1, is used to reliably generate the embedded key in the field and, therefore, is a potential attack target.

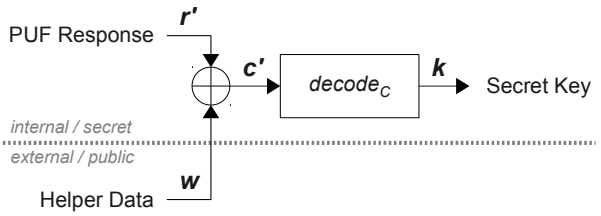


Fig. 1. Reconstruction module of a code-offset secure sketch

The key reconstruction of a code-offset secure sketch consists of the code-offset resolution of $\mathbf{r}'$, a noisy variant of the initial PUF response $\mathbf{r}$, and helper data $\mathbf{w}$ and the decoding function $decode_C$ of code C:

$$Rec(\mathbf{r}', \mathbf{w}) = decode_C(\mathbf{r}' \oplus \mathbf{w}) = \mathbf{k} \tag{1}$$

3.2 Helper Data Manipulation

Differential side-channel attacks require the manipulation of one of the function's inputs to generate changing intermediate (power) values. Since $\mathbf{r}'$ is determined by the unique PUF properties, manipulation of helper data $\mathbf{w}$ is the only way to achieve this. In the case of code-offset constructions, the manipulation of a single bit of $\mathbf{w}$ directly flips one input bit of the analyzed function $decode_C(\mathbf{r}' \oplus \mathbf{w})$, which is a desirable property for an attacker.

It is legit to assume deterministic write access to the helper data string for many embedded security applications because helper data is usually stored in external non-volatile memory, which, actually, is a profitable advantage of PUFs. However, if the helper data is located in a memory hard to manipulate, as it might be the case for smartcards, then this renders DPA on PUF key generation impossible or at least extremely time consuming.

In order to apply correlation-based DPA to the error correction of a code-offset secure sketch, first, an intermediate value of $decode_C()$ has

to be chosen as an attack point[3]. Then, power traces $\mathbf{t}_1, ..., \mathbf{t}_W$ with T samples have to be collected for W different helper data input vectors $\mathbf{w}_1, ..., \mathbf{w}_W$ resulting in a $W \times T$ trace matrix $\mathbf{T}$. Based on the chosen power model, for each of the R possible PUF response vectors $\mathbf{r}_1, ..., \mathbf{r}_R$, hypothetical intermediate values for each helper data vector $\mathbf{w}_1, ..., \mathbf{w}_W$ have to be calculated and stored as a $W \times R$ hypothesis matrix $\mathbf{H}$. Finally, the correlation between the T sample columns of $\mathbf{T}$ and each of the R PUF response hypothesis columns of $\mathbf{H}$ has to be computed [19] to obtain a $R \times T$ correlation matrix $\mathbf{M}$. The maximum correlation value in $\mathbf{M}$ gives the trace sample and the best correlating PUF bit vector hypothesis. This PUF bit vector represents the extracted secret which, together with the original helper data $\mathbf{w}$, can be used to calculate the embedded key $\mathbf{k}$.

For practical attacks, the noise contained in $\mathbf{r}'$ now and then changes bits of the ECC decoder input of a secure sketch, which makes the DPA inaccurate. Therefore, PUF noise generally leads to an increased number of required power traces.

3.3 DPA on Secure Sketch FPGA Implementation

Our secure sketch implementation used for the following DPA attack is shown in Figure 2. The decoding module incorporates a concatenation of an $(n{=}7, k{=}1, t{=}3)$ repetition code and an $(n{=}127, k{=}64, t{=}10)$ BCH code. Concatenated codes have been shown to achieve strong and efficient error correction implementations for PUF key generation [1,18]. We chose this combination to achieve a BCH code output error probability of less than 10^{-6} [1].

We embedded a 128-bit key into 1778 PUF response bits, which results in 1778 bits of helper data. Our implementation has two 7-bit input interfaces for chunks of PUF response bits and helper data bits. The code-offset XOR and the repetition decoding are implemented in combinational logic, which yields a decoded 1-bit output for each 7-bit helper data input word. After each repetition decoding, the output bit is shifted into the BCH decoder [11], which bit-serially decodes 127 input bits to a stable 64-bit word. This procedure is performed twice to obtain a 128-bit key.

We did not use a real PUF implementation, but provide PUF response bits as well as helper data from a preloaded circular buffer. This is important to obtain DPA results which are not influenced by a PUF's specific noise characteristics.

[3] Note that we assume that an attacker knows the function $decode_C()$ or has determined its characteristics by reverse engineering.

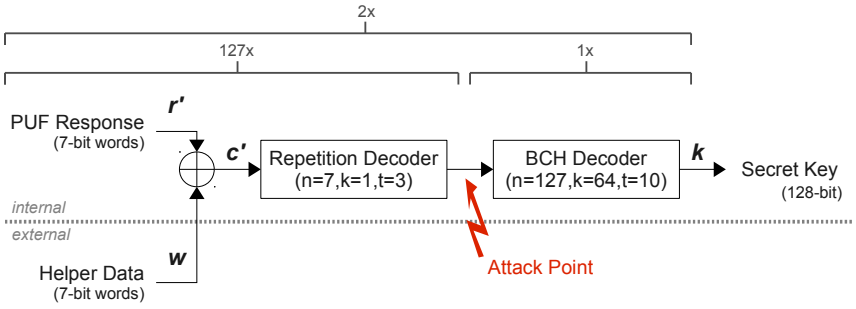


Fig. 2. Secure sketch implementation under test

We chose the output of the repetition code decoder, as shown in Figure 2 which is stored in the input register of the BCH decoder, as the intermediate value to attack. One reason therefore is the simple calculation of a repetition code decoding compared to the calculation of intermediate values of a BCH code. The second reason is that all codeword inputs of the 7-to-1 repetition decoder can be covered by manipulating the 7-bit helper data chunks, which leads to $W = 128$ traces per repetition codeword. We used a Hamming distance model hypothesis between two succeeding repetition code output bits to estimate the hypothetical power consumption under $W = 128$ different helper data manipulations and $R = 128$ possible PUF response bit vectors. For the first decoded bit, the preceding register value is assumed to be zero after reset.

We synthesized our design for a Xilinx XC3S200 FPGA and analyzed its power consumption over a 10 Ohm shunt resistor with a differential probe connected to a digital storage oscilloscope. We recorded 128 traces per repetition codeword for every possible manipulation of the 7-bit helper data input. We focused our analysis on the first four cycles after each helper data word was provided to the key reconstruction, because the repetition decoding happens directly afterwards. Afterwards, we correlated the hypothetical power values with the measured traces. For all following hypothesis correlation figures, we only depicted positive correlations because the linearity property of the analyzed circuit leads to the fact that inverted PUF inputs show a 'mirrored' negative correlation, which does not provide further information.

In Figure 3, the maximum correlation for all 7-bit PUF hypothesis are shown. The values were generated with only 128 measured traces without any preprocessing. The estimated significance bound [19] for 128

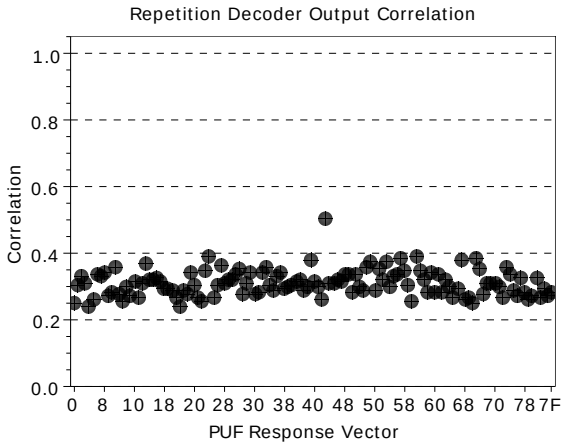


Fig. 3. Maximum correlation of repetition decoder output

traces is approx. $4/\sqrt{128} = 0.35$. Therefore, we are confident that our attack worked correctly, since the results show a correlation of 0.52 for the correct PUF bits (0x43) and leave all other hypothesis below or only slightly above the significance bound (0.38, in the best case).

For a full attack on all 1778 PUF bits of our implementation, an attacker would have to perform the described analysis for all 254 repetition code decoder runs, which results in a minimum of $254 \times 128 = 32\,512$ traces to record and analyze.

4 Codeword Masking

A popular method to prevent DPA attacks is masking input values of an algorithm with random masks [3,8,19] to destroy the dependency of intermediate and input values. However, this is not possible for ECCs because a random mask would be regarded as a random error vector introducing (additional) errors to the original codeword. Therefore, trivial masking cannot be applied without loosing the ECC's essential error correction properties.

In this section, we propose a codeword masking scheme which can be applied to linear ECCs without having an impact on their error correction capabilities and therefore not influencing the functionality of the original application, e.g. PUF key generation. We do not focus on non-linear ECCs because, in the vast majority of cases, PUF key generation is based on linear ECCs. We first explain the principle of codeword masking, show

how to apply it to a secure sketch architecture and then demonstrate its practical protection.

The linearity property [17] of the functions $encode_C$ and $decode_C$ of a linear code C defines, that the sum (XOR) of two codewords of C always results in another codeword of C. This also holds for concatenations of linear codes and represents the basis for our codeword masking scheme. We propose to mask an original codeword $\mathbf{c}$ of C by XORing it with the codeword mask $\mathbf{c_m}$, where $\mathbf{c_m}$ is a random codeword of C. The result of this boolean masking can still be decoded by $decode_C$ to cancel bit errors, if present. Decoding $\mathbf{c_m}$ leaves the random bit vector $\mathbf{m}$, which corresponds to the 'raw' mask. This mask can then be used to demask the decoded result by XORing:

$$decode_C(\mathbf{c} \oplus \mathbf{c_m}) = decode_C(\mathbf{c}) \oplus decode_C(\mathbf{c_m}) = decode_C(\mathbf{c}) \oplus \mathbf{m} \quad (2)$$

The relation between the original input codeword and the processed intermediate values is broken by this method while preserving the ECCs error correction features. Therefore, we propose it to protect linear ECCs from DPA attacks like the one shown in Section 3.

Looking at the overhead of codeword masking, the straight forward approach would be duplicating the decoding module. However, we propose to generate a random mask $\mathbf{m}$ (used for demasking) and encode it to obtain $\mathbf{c_m}$. Thereby, only the encoding function $encode_C$ has to be implemented, which has a significantly lower implementation complexity than the decoding function $decode_C$ [17]. Further, the encoding function is also required for the enrollment phase, which means that it might be implemented anyway and can be reused for masking purposes resulting in almost no overhead.

4.1 Secure Sketch Protection

The application of the proposed technique to secure sketches is shown in Figure 4. There, the helper data $\mathbf{w}$ is masked by the masking codeword $\mathbf{c_m} = encode_C(\mathbf{m})$ of a random mask $\mathbf{m}$. The resulting ECC decoder input can be decoded to $\mathbf{k} \oplus \mathbf{m}$, which results in the generated key $\mathbf{k}$ after demasking with the raw mask $\mathbf{m}$.

For a practical evaluation, we extended the secure sketch implementation described in Section 3 by the codeword masking scheme shown in Figure 4. In detail, we added a 128-bit mask register (same size as key), which is first encoded by the BCH code encoder and then by the repetition code encoder. We preloaded the mask register with random bits

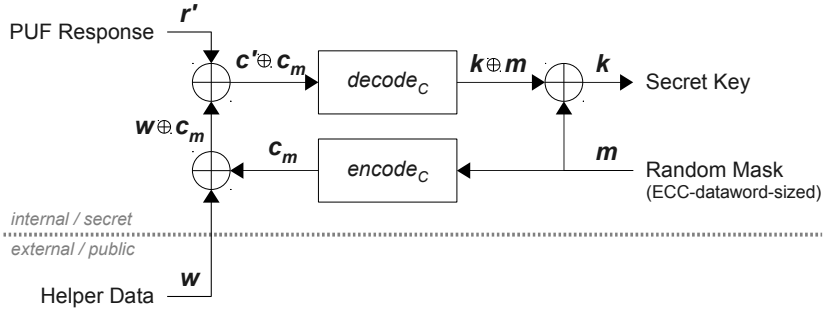
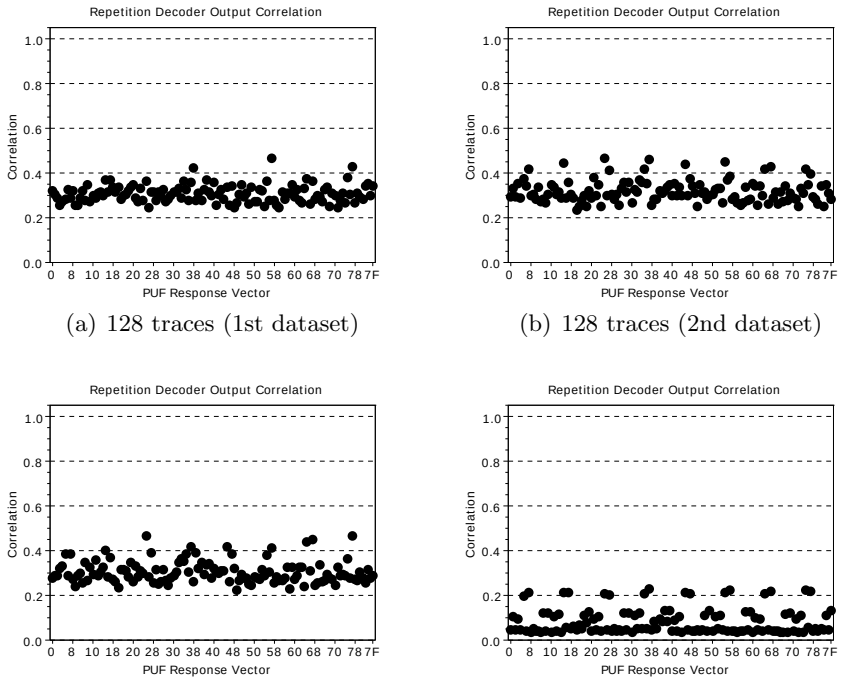


Fig. 4. Masked secure sketch (*Rec*)



(a) 128 traces (1st dataset)

(b) 128 traces (2nd dataset)

(c) 128 traces (3rd dataset)

(d) 6400 traces

Fig. 5. Maximum correlation of masked repetition decoder output

before each analysis run. The overhead in this proof-of-concept FPGA design sums up to 20% more slices, 45% more registers and 22% more look-up tables, which is mainly determined by the additional registers for the raw and the encoded codeword mask, but also by the additional BCH encoder. We regard these results as rather efficient, because the overhead is significantly below numbers for AES, where masked FPGA implementations are reported to have around 60% slice overhead [12]. Also, the overhead can be reduced even more, if the existing BCH encoder of the enrollment module is used for masking purposes.

We performed the same correlation-based first-order DPA attack on the repetition decoder output as described in Section 3 to practically evaluate the codeword masking countermeasure.

Figures 5(a), 5(b) and 5(c) show results for attacks on three different sets of 128 traces covering all 128 possible helper data manipulations. They exhibit significant correlations, but neither for the correct PUF response vector (0x43), nor for the same value at all. An attack based on 6400 traces (50 repetitions of the 128 helper data manipulations) leads to the correlations shown in Figure 5(d). Although 6400 traces do not sound like a high number for DPA, note that this attack only targets 7 of 1778 bits and, if successful, already a total of $254 \times 6400 = 1\,625\,600$ traces would be required for a complete attack. However, also with 6400 traces, the correct PUF value is not distinguishable. We blame the clearly visible patterns of significant correlations on the linearity of the repetition code implementation, but do not see them as a weakness of codeword masking because they do not reveal the secret PUF bits. This demonstrates the protective capabilities of the proposed masking scheme for PUF error correction against first-order DPA attacks. Higher-order attacks might be a threat to it, but were out-of-scope for this work.

5 Wider Scope for Codeword Masking

This section widens the scope of codeword masking to fuzzy extractors and non-code-offset key generation. Also, it can be applied to any other application of linear ECCs.

5.1 Code-Offset Fuzzy Extractors

A secure sketch has to be extended to a fuzzy extractor including a randomness extractor *Ext* to obtain a full-entropy key $\mathbf{k}$, if a PUF's output is not completely random.

For code-offset fuzzy extractors, the secret key $\mathbf{k}$ is not encoded to codeword $\mathbf{c}$, but extracted from PUF response $\mathbf{r}$ by a randomness extractor *Ext*. The codeword $\mathbf{c}$ is chosen at random and only serves for code-offset error correction. During the reproduction procedure *Rep*, see Figure 6, the noisy codeword $\mathbf{c'} = \mathbf{r'} \oplus \mathbf{w}$ is corrected to $\mathbf{c} = \mathit{correct}_C(\mathbf{c'})$ and then again XORed with the helper data $\mathbf{w}$ to obtain the initial PUF response vector $\mathbf{r} = \mathbf{c} \oplus \mathbf{w}$.

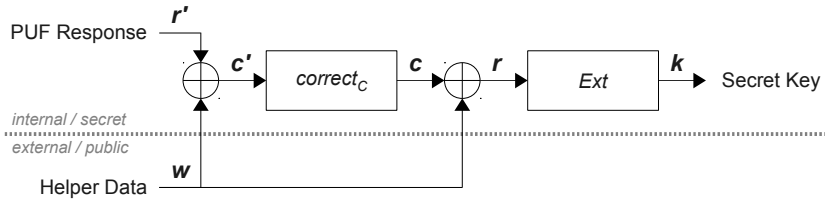


Fig. 6. Code-offset fuzzy extractor (*Rep*)

Since the error correction of the code-offset fuzzy extractor architecture is very similar to the one of a secure sketch, it is vulnerable to the same attack as shown in Section 3. Additionally, the second processing of helper data $\mathbf{w}$ allows to manipulate intermediate values of the extractor module *Ext* and, thereby, to mount extractor DPA attacks based on helper data manipulations. The double influence of helper data $\mathbf{w}$ is shown in the formal equation of a code-offset fuzzy extractor's reproduction phase:

$$\mathit{Rep}(\mathbf{r'}, \mathbf{w}) = \mathit{Ext}(\mathit{correct}_C(\mathbf{r'} \oplus \mathbf{w}) \oplus \mathbf{w}) = \mathbf{k} \tag{3}$$

In previously proposed fuzzy extractor implementations [1,18], Toeplitz hashing [16] is chosen as an efficient randomness extractor. In this linear hash algorithm, the state of a Linear Feedback Shift Register (LFSR) is XORed into an accumulator if the input bit is 1. Otherwise, the LFSR will only be shifted without XORing. In the end, the hash value is a linear combination of LFSR states, selected by the input data.

With the proposed codeword masking scheme, it is possible to consistently mask such fuzzy extractor architectures. The application to the error correction module is identical as for secure sketches, but then the masked PUF response $\mathbf{r} \oplus \mathbf{c_m}$ is fed into the Toeplitz hashing *TH* as shown in Figure 7. The encoded mask is also processed by the Toeplitz hashing. This leads to a hashed masking codeword $\mathit{TH}(\mathbf{c_m})$ which represents the

correct mask for the masked key $TH(\mathbf{r} \oplus \mathbf{c_m})$ because of the linearity property of Toeplitz hashing:

$$TH(\mathbf{r} \oplus \mathbf{c_m}) = TH(\mathbf{r}) \oplus TH(\mathbf{c_m}) = \mathbf{k} \oplus TH(\mathbf{c_m}) \tag{4}$$

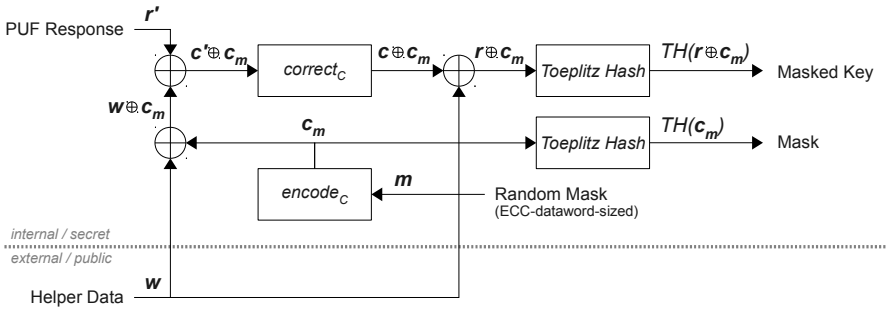


Fig. 7. Masked code-offset fuzzy extractor (*Rep*)

The resulting pair of masked key and corresponding mask can be used to resolve the masking, if required, but can also be provided to the subsequent crypto module leading to a consistently masked code-offset fuzzy extractor from the first helper data input to the finally cryptographic algorithm. Note that implementing such an architecture also protects devices from the SPA attacks on fuzzy extractors shown in related work [13,22,20]

5.2 Robust Sketches and Robust Fuzzy Extractors

In 2005, Boyen et al. [2] explained that an active adversary can gain information about the user's biometric (or PUF) by maliciously manipulating the communication between a server (implementing a fuzzy extractor) and the user. As a solution, they proposed robust sketches and robust fuzzy extractors. However, we explain that these constructions do not protect against physical attackers.

The proposed robust constructions use a hash function H to generate a hash value $\mathbf{h} = H(\mathbf{k}, \mathbf{w})$ which is stored along with helper data $\mathbf{w}$. During each reconstruction, a (maybe manipulated) secret key $\hat{\mathbf{k}}$ is generated as in a standard secure sketch or fuzzy extractor, but before using $\hat{\mathbf{k}}$, the hash value $\hat{\mathbf{h}} = H(\hat{\mathbf{k}}, \hat{\mathbf{w}})$ is compared to the previously generated value $\mathbf{h}$.

If the hash values match, $\hat{\mathbf{w}}$ was not manipulated and $\hat{\mathbf{k}} = \mathbf{k}$, otherwise, $\hat{\mathbf{k}}$ will be discarded.

While this construction is secure for remote scenarios, we want to stress that it does not hold for a physical attacker, who is able to observe side-channel leakage. The reason for that is, that the operations of standard secure sketches and fuzzy extractors, which cause exploitable side-channel leakage, still need to be performed *before* a decision can be made if the helper data was manipulated or not. Therefore, observing side-channels enables an attacker to extract information about intermediate results, even before a robust sketch or a robust fuzzy extractor detects the manipulation.

5.3 Masking Other PUF Key Generation Algorithms

The main part of this contribution is based on code-offset algorithms, however, we want to emphasize that the demonstrated DPA attack as well as the proposed codeword masking are also applicable to other PUF key generation algorithms, e.g., Index-Based Syndrome coding (IBS) [25] and its extension Complementary IBS (C-IBS) [10].

IBS stores indices of reliable PUF output bits as helper data, which is used to select the most reliable bits during reconstruction. C-IBS additionally stores indices of reliable bits with complementary value to increase reliability. Both constructions are usually supported by an ECC implementation to achieve lower residual error probabilities.

Regarding DPA attacks, there is a slight difference compared to code-offset algorithms. While flipping a bit in code-offset helper data directly leads to a flipped bit of the ECC input data, for IBS, an attacker has to guess (from a small number of choices) a valid index of an inverted PUF bit to achieve the necessary bit flip. For C-IBS, an attacker has an easy job again, since exchanging original and complementary indices in the helper data deterministically causes a bit flip.

In order to protect the error correction module of IBS and C-IBS implementations, we propose to apply random codeword mask to PUF bits selected by IBS/C-IBS before processing them by ECCs. Afterwards, the masked key can be demasked or forwarded to the subsequent crypto module.

6 Conclusion

We showed that vulnerabilities of error correction modules used in PUF-based key generation can be exploited by DPA and experimentally demon-

strated its feasibility on power traces of a secure sketch implementation. In order to protect PUF-based key generation, we proposed a codeword masking scheme based on random codewords, which maintains an ECC's full error correction capability. We verified our approach by attacking a masked secure sketch implementation. Further, we showed how codeword masking can be used to consistently mask fuzzy extractors from the first code-offset to the point where a crypto module uses the generated key.

Our results show that the proposed masking scheme is an important step towards side-channel attack resistance of PUF key generation algorithms. Besides code-offset-based key generation, also other key embedding algorithms like IBS and C-IBS can benefit from protecting their ECC implementations by codeword masking. Even non-PUF applications, such as, reliable memories or the McEliece cryptosystem, can use codeword masking as a valuable countermeasure against shown side-channel vulnerabilities of their ECC implementations.

Acknowledgements

The authors would like to thank Johann Heyszl, Benedikt Heinz, Dieter Schuster, Matthias Hiller and Marc Stöttinger for helpful discussions.

References

1. C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In *CHES '08: Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 181–197, Berlin, Heidelberg, 2008. Springer-Verlag.
2. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 561–561. Springer Berlin / Heidelberg, 2005.
3. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology - CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 791–791. Springer Berlin / Heidelberg, 1999.
4. J. Dai and L. Wang. A study of side-channel effects in reliability-enhancing techniques. In *Proceedings of the 2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, DFT '09, pages 236–244, Washington, DC, USA, 2009. IEEE Computer Society.
5. S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and implementation of PUF-based "unclonable" RFID ICs for anti-counterfeiting and security applications. In *RFID, 2008 IEEE International Conference on*, pages 58–64, 2008.

6. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer Berlin / Heidelberg, 2004.
7. T. Esbach, W. Fumy, O. Kulikovska, D. Merli, D. Schuster, and F. Stumpf. A new security architecture for smartcards utilizing PUFs. In *Proceedings of the 14th Information Security Solutions Europe Conference (ISSE'12)*. Vieweg+Teubner Verlag, 2012.
8. L. Goubin and J. Patarin. DES and differential power analysis the "duplication" method. In . Ko and C. Paar, editors, *Cryptographic Hardware and Embedded Systems*, volume 1717 of *Lecture Notes in Computer Science*, pages 728–728. Springer Berlin / Heidelberg, 1999.
9. J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In P. Paillier and I. Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
10. M. Hiller, D. Merli, F. Stumpf, and G. Sigl. Complementary IBS: Application specific error correction for PUFs. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, pages 1 –6, june 2012.
11. E. Jamro. The design of a vhdl based synthesis tool for bch codecs. Master's thesis, School of Engineering, The University of Huddersfield, Sep 1997.
12. N. Kamoun, L. Bossuet, and A. Ghazel. SRAM-FPGA implementation of masked s-box based DPA countermeasure for AES. In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 74–77, 2008.
13. D. Karakoyunlu and B. Sunar. Differential template attacks on PUF enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1 –6, dec 2010.
14. P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, 1996. Springer-Verlag.
15. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
16. H. Krawczyk. LFSR-based hashing and authentication. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–139, London, UK, 1994. Springer-Verlag.
17. S. Lin and D. J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.
18. R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In C. Clavier and K. Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2009.
19. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
20. D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Side-channel analysis of PUFs and fuzzy extractors. In *4th International Conference on Trust and Trustworthy Computing (TRUST2011)*, Pittsburgh, PA, USA, June 2011. Springer.
21. H. Molter, M. Stöttinger, A. Shoufan, and F. Strenzke. A simple power analysis attack on a McEliece cryptoprocessor. *Journal of Cryptographic Engineering*, 1:29–36, 2011.

22. D. Schuster. Side-channel analysis of physical unclonable functions (PUFs). Diploma thesis, Technische Universität München, Dec. 2010.
23. F. Strenzke, E. Tews, H. Molter, R. Overbeck, and A. Shoufan. Side channels in the McEliece PKC. In J. Buchmann and J. Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 216–229. Springer Berlin / Heidelberg, 2008.
24. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 9–14, 2007.
25. M.-D. M. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Des. Test*, 27(1):48–65, 2010.