Eliminating Leakage in Reverse Fuzzy Extractors

André Schaller
Security Engineering Group
Technische Universität Darmstadt
and CASED
Darmstadt, Germany
Email: schaller@seceng.informatik.tu-darmstadt.de

Boris Škorić
SEC group
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: b.skoric@tue.nl

Stefan Katzenbeisser
Security Engineering Group
Technische Universität Darmstadt
and CASED
Darmstadt, Germany
Email: katzenbeisser@seceng.informatik.tu-darmstadt.de

***Abstract*—In recent years Physically Unclonable Functions (PUFs) have been proposed as a promising building block for security related scenarios like key storage and authentication. PUFs are physical systems and as such their responses are inherently noisy, precluding a straightforward derivation of cryptographic key material from raw PUF measurements. To overcome this drawback, Fuzzy Extractors are used to eliminate the noise and guarantee robust outputs. A special type are Reverse Fuzzy Extractors, shifting the computational load of error correction towards a computationally powerful verifier. However, the Reverse Fuzzy Extractor reveals error patterns to any eavesdropper, which may cause privacy issues (if the PUF key is drifting, the error pattern is linkable to the identity) and even security problems (if the noise is data-dependent and multiple protocol transcripts can be linked to the same user). In this work we investigate this leakage and propose a modified protocol that eliminates the problem.**

I. Introduction

In the past decade Physically Unclonabe Functions (PUFs) have attracted increasing attention. With their desirable property of unclonability they were proposed as a promising security building block that could be applied to various identification and authentication applications. Since then several protocols featuring PUFs have been proposed, such as key storage, authentication and remote attestation schemes [1]–[3]. In this paper we focus on key storage. This application is sometimes referred to as 'Physically Obfuscated Key'. In particular, we will consider the use of a PUF key in the context of privacy-preserving protocols that are designed to hide the identity of the prover from eavesdroppers.

PUFs are physical systems and thus their measurements always contain a certain amount of noise. However, cryptographic primitives like hashes and ciphers do not tolerate any noise. Thus, the noise in a PUF measurement must be dealt with before the measurement can be given as input to a cryptographic primitive. This introduces a complication: redundancy data (for the error correction) needs to be stored somewhere as part of the PUF enrollment data. The usual attacker model states that enrollment data is always visible to the adversary. Hence, error correction needs to be designed such that the redundant data hardly leaks information about the PUF key. An error correction scheme that satisfies this requirement is variously known as Helper Data Scheme (HDS), Secure Sketch (SS) or Fuzzy Extractor (FE). FEs have the additional property that they generate a (nearly) uniform key. A FE can be trivially derived from a SS. One of the most popular HDSs is the Code Offset Method that employs a linear Error-Correcting Code (ECC). Particularly compact implementations are possible if *syndrome decoding* is used.

In many PUF applications the prover is assumed to be a resource-constrained device. In the key reconstruction phase the prover needs to perform an ECC decoding step, which may be infeasible given the constraints. An elegant solution was proposed in [4], where it was detailed how the ECC decoding can be securely outsourced to the verifier. The scheme was dubbed 'Reverse Fuzzy Extractor'. The most difficult HDS task for the prover is now merely to compute a syndrome, which can be done very efficiently in hardware. On the downside, in each protocol run the Reverse FE reveals to eavesdroppers which error pattern is present in the PUF measurement as compared to the enrollment measurement. It was argued in [4] that the PUF key is secure as long as the measurement noise is independent of the PUF value itself.

In this paper we argue that this assumption is likely not fulfilled by many practical PUF realizations. Our results are based on the observation that PUF instances and thus their measurements exhibit a 'drift' in the values of PUF response bits. This drift is a function of time and it is characteristic to individual PUF instances. Considering this drift an attacker might be able to identify PUF instances by analyzing the error pattern that is communicated between client and server, since the drift is directly reflected in the error pattern. This creates a *privacy* risk, in particular if the PUF is used in privacy-preserving protocols.

Furthermore, once the adversary is able to distinguish between different PUF devices, the revealed error patterns may cause a *security* risk on top of the privacy issue. This risk exists if the probability distribution of the noise is data-dependent; then the noise reveals information about the PUF key.

In this work we are elaborating on both issues – the privacy risk of linking error patterns to individual PUF devices, and the security risk of error patterns leaking information about PUF keys – by analyzing real PUF measurements. In particular, we show that several PUF types indeed show a drift, which can be modeled. Subsequently, we propose a protocol that improves the Reverse Fuzzy Extractor and eliminates security and privacy risks stemming from this drift.

The rest of the paper is organized as follows. In Section II we give a brief overview on PUFs and Fuzzy Extractors. Section III defines notations and introduces the Reverse Fuzzy

Extractor. The problem statement is explained in Section IV. In Section V we evaluate our assumptions on real data. We introduce an improved version of the Reverse Fuzzy Extractor protocol in Section VI. Lastly, we conclude our work in Section VII.

II. Related Work

A. Physically Unclonable Functions

A Physically Unclonable Function (PUF) is a complex physical structure that generates a response to a pysical stimulus. The response depends on the challenge as well as on the micro- or nanoscale physical structure of the PUF itself. One typically assumes that the PUF can not be cloned, not even by the manufacturer of the device. Furthermore, the challenge-response behavior of the physical system is assumed to be complex enough such that the response to a given challenge can not be predicted.

Several different PUF constructions exist; for an overview we refer to [5]. Among them are memory-based PUFs, such as SRAM PUFs, which exploit biases in memory cells. At the power-up phase these cells initialize to either the value of zero or one. Most cells show a significant tendency to initialize to one of both values. The entirety of the the start-up values creates a start-up pattern, which is taken as PUF response. PUFs can also exploit random timing characteristics of circuits, among them ring oscillator PUFs and Arbiter PUFs.

Due to physical characteristics of the device, measurments of a PUF response are subject to noise; thus, subsequent measurements will be slightly different. In order to use them in cryptographic protocols, a stable response must be generated. This is done by employing a Fuzzy Extractor [6], which extracts the stable part of the PUF response and transforms it to a uniformly distributed value.

B. Fuzzy Extractors

The authors of [6] introduced Fuzzy Extractors as a means to deal with the noise. Commonly, Fuzzy Extractors work in two phases, a generation phase $\texttt{Gen}()$ performed upon enrollment and a reconstruction phase $\texttt{Rec}()$ performed after each measurement. During $\texttt{Gen}()$, a secret key K and a public Helper Data W are derived from a noisy PUF reference (enrollment) measurement X. The algorithm $\texttt{Rec}()$ transforms a noisy PUF measurement X' back into the key K, thereby using the Helper Data W. This works as long as X and X' are close enough (e.g. are two PUF responses to the same challenge). Usually the reconstruction is achieved using an error correcting code.

III. Preliminaries

A. Notation

The notation 'log' stands for the base-2 logarithm. Random variables are written in capital letters and their values in lowercase. The binary entropy function is written as

$$h(p) \stackrel{\text{def}}{=} -p \log p - (1-p)\log(1-p). \tag{1}$$

The Shannon entropy of a random variable X is denoted as $\mathsf{H}(X)$, and mutual information as $I(X;Y)$.

B. The Reverse Fuzzy Extractor

We briefly review the Reverse Fuzzy Extractor protocol [4].[1] We omit all details that are not critical for the key reconstruction itself (i.e., signal processing of the raw PUF data, additional protection of the helper data, hashes of the key, quantities derived from the key, usage of the key, etc.). The description below is identical to the 'Syndrome-Only' Code Offset Method [7] with the sole difference that syndrome decoding is outsourced to the verifier.

System setup:
The parties agree on a linear error correcting code $\mathcal{C}$, with message length k and codeword length n. The encoding algorithm is $\texttt{Enc} : \{0,1\}^k \to \{0,1\}^n$, and the algorithm for computing the syndrome is denoted as $\texttt{Syn} : \{0,1\}^n \to \{0,1\}^{n-k}$. The code is chosen such that an efficient syndrome decoder $\texttt{SynDec} : \{0,1\}^{n-k} \to \{0,1\}^n$ exists. The parties also agree on a key derivation function $\texttt{KeyDeriv} : \{0,1\}^n \to \{0,1\}^\ell$.

Enrollment:
A PUF enrollment measurement $X \in \{0,1\}^n$ is obtained. The helper data $W = \texttt{Syn}(X)$ is computed. The prover stores W, while the verifier stores $K = \texttt{KeyDeriv}(X)$.

Reconstruction:
1) The prover performs a fresh measurement $X' \in \{0,1\}^n$. He computes $\Sigma = W \oplus \texttt{Syn}(X')$ and sends Σ to the verifier.

2) The verifier computes the error pattern $E = \texttt{SynDec}(\Sigma)$ and sends E to the prover.

3) The prover computes the estimators $\hat{X} = X' \oplus E$ and $\hat{K} = \texttt{KeyDeriv}(\hat{X})$.

Note that this protocol is extremely lightweight, as the prover only has to perform one $\texttt{Syn}$ and one $\texttt{KeyDeriv}$ operation. Note further that $\Sigma = \texttt{Syn}(X \oplus X')$ due to the linearity of the code $\mathcal{C}$. Hence, if there is not too much noise, E is the error pattern that maps X' back to X. Also note that E does not leak anything about X *only if the noise is independent of the data*.

IV. Problem Statement

The Reverse Fuzzy Extractor protocol as described in [4] and Section III-B has two potential problems.

If the PUF noise is not independent of the measurement X, then some information about X is leaked to eavesdroppers via the error pattern E, communicated in step 2 of the reconstruction. For instance, imagine that for a single bit of the PUF response a $0 \to 1$ transition is much more likely than a $1 \to 0$ transition. Then the error locations in E point to locations where a zero in X is much more likely than a one. This is a *security* risk. It will become a serious threat if the adversary observes multiple transcripts from the same prover, carrying different information about X, and is able to link those transcripts together. Note that this problem is only mitigated by construction if the PUF is used in *privacy-preserving identification* protocols (i.e., protocols in which an eavesdropper does not learn the PUF identity). In this case the adversary cannot link different protocol transcripts to the same PUF so that it is difficult for him to properly combine the

[1] We actually describe a more general primitive: a Secure Sketch.

different pieces of information that he has. For other protocols we show in the next section that this security risk is present.

Another problem arises from the fact that biases in bits of the PUF response tend to drift over time as the PUF ages. This can lead to error patterns that are unique for each PUF. Even if there is no leakage about X, the error pattern E may leak the *identity* of the PUF. If the prover is using the PUF to run a privacy-preserving identification protocol (as opposed to authentication), then this identity leakage is a *privacy* risk. If this problem is not solved, it is impossible to use the Reverse Fuzzy Extractor for privacy-preserving identification.

Note that there is an interaction between the two problems: the identity leakage makes it easier for the adversary to exploit the key leakage from the error patterns.

V. Evaluation of the Leakage

Some PUF instances show a bias of individual response bits towards either zero or one. In this section we show that these biases change over time, which we call the drift of a PUF. Furthermore, we provide a model for the drift and estimate the information leakage due to the drift in Reverse Fuzzy Extractors.

A. Drift model

We adopt the bias-based PUF model proposed in [8]. We refer to this work for a detailed explanation. In brief, we denote the bias of a PUF resonse bit i during enrollment as a value $b_i \in [0,1]$; it can be estimated by counting the number x_i of occurrences of a '1' bit at position i during k enrollment measurements: $b_i = x_i/k$. A PUF is fully characterized by a vector of biases, $b = (b_i)_{i=1}^n$. Similarly, b'_i represents the bias of bit i at a later time. It can be estimated from the number x'_i of '1' bits in position i in a series of l PUF responses: $b'_i = x'_i/l$.

Finally, we model the drift by a transition probability $\tau(b'|b)$ indicating how likely it is that the PUF has bias vector b' at a later time given that it had bias vector b at enrollment. Assuming that the n PUF response bits are mutually independent (this assumption seems justified as we did not see any correlation between response bits in the PUF types under investigation), and that drift behavior is the same for all bits, we can express the transition probability for the entire PUF as

$$\tau(b'|b) = \prod_{i\in[n]} \tau_0(b'_i|b_i). \tag{2}$$

The function τ_0 does not depend on the bit index i. To estimate τ_0 we made single-bit histograms of drifted bit biases, conditioned on the enrolled bit bias, i.e., for each possible value of b_i we made a histogram counting $b_i \to b'_i$ occurrences. Here the $b_i \to b'_i$ transitions were collected from all bits. Finally we converted the histograms to probability distributions.

B. Drift results

We made use of PUF measurement data obtained in the UNIQUE project[2]. During this project custom ASICS with different PUF types, including SRAM, latch, D-Flip-Flop (DFF), Arbiter and ring oscillator PUFs, were developed and tested under different conditions. The UNIQUE data set includes PUF measurements of devices which were exposed to an accelerated aging process. The simulation of aging is based on the Negative Bias Temperature Instability (NBTI) mechanism, carried out by by operating the ASICs at an extreme temperature of $+85\,^{\circ}\mathrm{C}$ and with high supply voltage of $1.44V$ (120% of the $1.2V$ standard V_{dd}). The treatment lasted for 2150 hours corresponding to an aging factor of 18.2. This way, continuous use of the PUF-enhanced device can be simulated in short time.

Three different datasets were available for our experiments: enrollment data taken right after manufacturing (referred to as time t_0), measurements at the beginning of the aging process (at time t_1) and measurements after the aging process had terminated (time t_2). Measurements of t_1 correspond to a simulated operating time of approximately 1 week with respect to t_0 whilst the dataset at t_2 contains measurements of devices with a simulated operating time of approximately 4.5 years. For our bias transition model we compared t_0 versus t_1 and t_0 versus t_2.

Figure 1 shows τ_0 for SRAM, latch, DFF and Ring Oscillator (RO) PUFs, computed between enrollment data and measurements taken at time t_1 ('beginning of aging') and t_2 ('end of aging'), respectively.

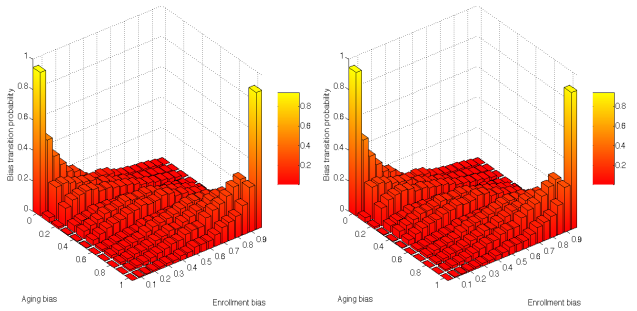
In Figures 1a, 1c and 1e we can observe a diagonal 'saddle' between $(0,0)$ and $(1,1)$ for the data taken at time t_1. This indicates that SRAM, latch and DFF PUFs show a stable behavior regarding their bits and keep the same bias over a short operating time. The RO PUF (see Figure 1g) is an exception, featuring an 'island' of high probabilities in the middle of the plot area, indicating more transitions to bias 0.5 (random behavior); this is expected, as ring oscillators can be used to generate random numbers as well.

After the aging process, we see a flattening of the 'saddle' for all PUF types (Figures 1b, 1d, 1f, 1h). This indicates, as expected, that there is significant drift after a longer operation time. Note that not all the transition probabilities are symmetric under $0 \leftrightarrow 1$ reversal; this phenomenon can mainly be observed for the latch and RO PUFs.

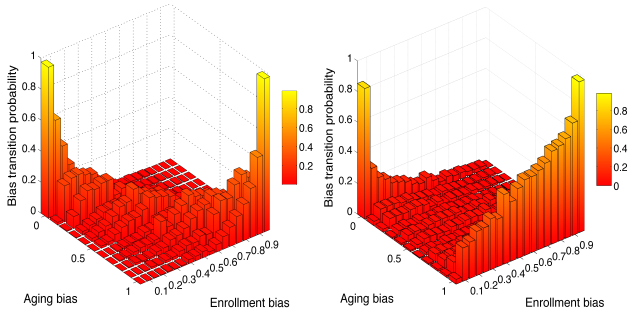
The FE reconstruction phase typically employs only a single measurement ($l = 1$). Hence, in practice FEs usually do not use fine-grained information about the biases to derive stable keys. Instead, fine-grained bias information is used only for the selection of reliable bits. A FE will typically store pointers to stable bits (i.e., bits that have an enrollment bias close to 0 or 1); only those response bits are then used for key derivation.

In Fig. 2 we show the aging effect on bits with enrollment bias $b_i \in [0, 0.05] \cup [0.95, 1]$. The vertical axis depicts the bit error rate, considering only those bits. Again, we compute the error rate between enrollment measurements (t_0) and data obtained at times t_1 or t_2. In the boxplots the red line in each box indicates the median bit error rate. The colored bottom and top of each box marks the 25^{th}/75^{th} percentile of the bit error rate. The height of the boxes display the inter quartile range (IQR). The whisker's ends indicate the lowest and highest bit

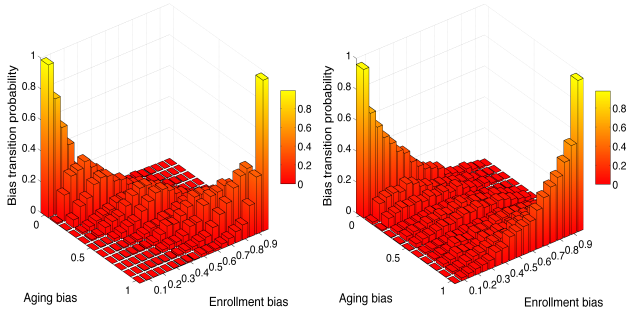
[2] http://www.unique-project.eu



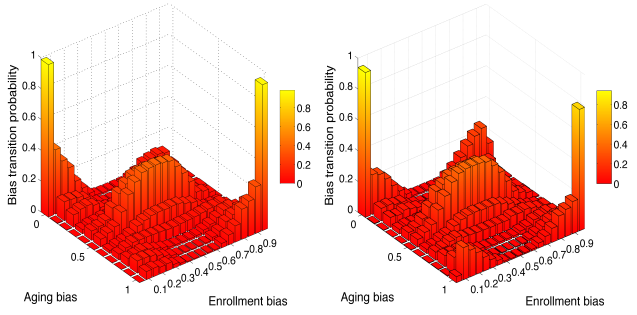
(a) SRAM: beginning of aging (b) SRAM: end of aging



(c) Latch: beginning of aging (d) Latch: end of aging



(e) DFF: beginning of aging (f) DFF: end of aging



(g) RO: beginning of aging (h) RO: end of aging

Fig. 1. Bias transition probabilities $\tau_0(b'|b)$ between enrollment and beginning of aging (left) and end of aging (right) for SRAM, latch, DFF and ring oscillator (RO) PUFs.

error rates that are within 1.5 times the IQR of the box edges. Single outsider values are marked by red plus signs.

The results indicate that the $t_0 \to t_2$ bit error rate is higher for the SRAM and ring oscillator PUFs with respect to the $t_0 \to t_1$ bit error rate.

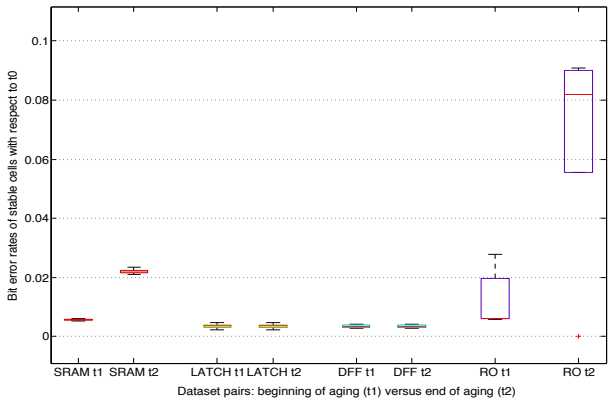


Fig. 2. Comparison of bit error rate (fractional Hamming Distance) between the enrollment measurements (t_0) the measurements from the beginning of the aging process (t_1), respectively the end of the aging process (t_2)

C. Leakage analysis

The results of Section V-A show that aging indeed causes drifting of the PUF measurement X' over time. Thus, the noise $E = X' \oplus X$ in the Reverse FE protocol contains a part $D \in \{0,1\}^n$ (the drift) that changes only over long time scales, while the rest of E consists of short-timescale random noise N unrelated to aging. We can represent E as $E = D \oplus N$.

In order to analyze leakage, we introduce a simplified drift model in which the biases are binarized to 0/1 values, and only reliable components are taken into account. The model has only two parameters: p_0, the probability of a $0 \to 1$ bit transition, and p_1, the probability of a $1 \to 0$ transition. The numerical values of these parameters slowly vary as a function of time. Table I shows the transition probabilities $0 \to 1$ and $0 \to 1$ of stable cells with enrollment biases $b_i \in [0, 0.05] \cup [0.95, 1]$. As expected the numbers show a significant imbalance for SRAM, DFF and latch PUFs with respect to the two transition probabilities.

We first quantify the privacy leakage of the Reverse FE protocol by observing the drift.

Lemma 1: Let X_1 and X_2 be the enrollment measurements of two different PUFs, uniformly distributed on $\{0,1\}^n$. Let D_1 and D_2 be their respective drifts. Let the drift be independent in each bit, with parameters p_0, p_1 as defined above. Then the Hamming distance between D_1 and D_2 is binomial-distributed, with parameters n and P_{uneq}, where

$$P_{\text{uneq}} = \frac{1}{2} - \frac{1}{2}(p_0 - p_1)^2. \tag{3}$$

Proof: In bit i we have the following conditional probabilities,

$$\begin{aligned} &\text{Prob}[D_{1,i} \neq D_{2,i} | X_1 = x_1, X_2 = x_2] \\ &= \begin{cases} 2p_0(1-p_0) \text{ if } x_{1,i} = x_{2,i} = 0 \\ 2p_1(1-p_1) \text{ if } x_{1,i} = x_{2,i} = 1 \\ p_0p_1 + (1-p_0)(1-p_1) \text{ if } x_{1,i} \neq x_{2,i} \end{cases} \end{aligned} \tag{4}$$

We compute $P_{\text{uneq}} \stackrel{\text{def}}{=} \text{Prob}[D_{1,i} \neq D_{2,i}] = \mathbb{E}_{x_1x_2}\text{Prob}[D_{1,i} \neq D_{2,i} | X_1 = x_1, X_2 = x_2] = \frac{1}{4}\sum_{x_{1,i}x_{2,i}} \text{Prob}[D_{1,i} \neq D_{2,i} | X_1 = x_1, X_2 = x_2]$. Performing the summation and then simplifying the result

TABLE I. TRANSITION PROBABILITIES $0 \to 1$ AND $0 \to 1$ FOR t_0 VS. t_1 AND t_0 VS. t_1 (BIASES $b_i \in [0, 0.05] \cup [0.95, 1]$. MEAN±STANDARD DEVIATION).

Transition	SRAM	LATCH	DFF	RING
		t_0 vs. t_1		
$0 \to 1$	0.0036 ± 0.0002	0.0025 ± 0.0007	0.0017 ± 0.0003	0.0076 ± 0.0055
$1 \to 0$	0.0020 ± 0.0001	0.0012 ± 0.0003	0.0019 ± 0.0004	0.0050 ± 0.0043
		t_0 vs. t_2		
$0 \to 1$	0.0130 ± 0.0004	0.0222 ± 0.0135	0.0062 ± 0.0028	0.0350 ± 0.0199
$1 \to 0$	0.0091 ± 0.0004	0.0041 ± 0.0038	0.0029 ± 0.0010	0.0323 ± 0.0184

yields (3). The drift in each bit position is independent; therefore the Hamming weight is the result of n independent events, each of which increments the Hamming weight with probability P_{uneq}.

Corollary 2: Let X_1 and X_2 be the enrollment measurements of two different PUFs, uniformly distributed on $\{0,1\}^n$. Let D_2 and D_2 be their respective drifts. Let the drift be independent in each bit, with parameters p_0, p_1. Then the expected Hamming distance between D_1 and D_2 is $\mu_{\text{HD}} = nP_{\text{uneq}}$, and the variance is $\sigma^2_{\text{HD}} = nP_{\text{uneq}}(1 - P_{\text{uneq}})$.

Proof: Follows from Lemma 1 and the properties of the binomial distribution.

If p_0 and p_1 are large enough, the drift D becomes a unique characterizing property for each PUF, as quantified in Corollary 2. Furthermore, if the short-timescale noise does not mask the drift, D is observable by the adversary through the error pattern E in the Reverse FE protocol transcripts (Section III-B): The adversary can observe multiple instances of an authentication protocol run by the same PUF device; by averaging he gets a good estimate of D.

Next we quantify the key leakage. Since we did not specify the `KeyDeriv` algorithm, we cannot compute the mutual entropy between D and the PUF key in general. Instead, we compute the mutual information between D and X.

Theorem 3: Let X be a PUF enrollment measurement, uniformly distributed on $\{0,1\}^n$. Let D be the drift. Let the drift be independent in each bit position, with parameters p_0, p_1. Then

$$\mathsf{H}(D) = nh(\frac{p_0+p_1}{2}), \quad (5)$$

$$I(D;X) = n\left[h(\frac{p_0+p_1}{2}) - \frac{h(p_0)+h(p_1)}{2}\right]. \quad (6)$$

Proof: Since all PUF bits are independent, we focus on a single (arbitrary) bit i. We have $\text{Prob}[D_i = 1] = \text{Prob}[X_i = 0]\text{Prob}[D_i = 1|X_i = 0] + \text{Prob}[X_i = 1]\text{Prob}[D_i = 1|X_i = 1] = (p_0+p_1)/2$. This yields $\mathsf{H}(D_i) = h(\frac{p_0+p_1}{2})$. Eq. (5) follows. Next we have

$$\begin{aligned}\mathsf{H}(D_i|X_i) &= \mathbb{E}_{x_i}\mathsf{H}(D_i|X_i = x_i)\\ &= \Pr[X_i = 0]\mathsf{H}(D_i|X_i = 0)\\ &\quad +\Pr[X_i = 1]\mathsf{H}(D_i|X_i = 1)\\ &= \frac{1}{2}h(p_0) + \frac{1}{2}h(p_1). \quad (7)\end{aligned}$$

We multiply $\mathsf{H}(D_i|X_i)$ by n and use $I(D;X) = \mathsf{H}(D) - \mathsf{H}(D|X)$.

Theorem 3 shows precisely how much the adversary learns about X from the observed D. Only if the drift direction is independent of X (parameters $p_0 = p_1$), then $I(D;X) = 0$, meaning that there is no leakage. In the case $p_1 = p_0 + \alpha$, with small $|\alpha| \ll 1$, a Taylor expansion of (6) gives $I(D;X) = \alpha^2/(8p_0(1-p_0)\ln 2) + \mathcal{O}(\alpha^3)$.

VI. SOLVING THE LEAKAGE PROBLEMS

In this section we present a modified Reverse Fuzzy Extractor in which the bias drift does not cause leakage. We use a combination of two ideas. First, the prover keeps track of the computed error patterns E over time. If E starts to exhibit behavior constant in time (a drift D), then the prover device modifies its stored helper data in such a way that the drift is compensated; future error patterns E will thus not reveal the drift.

Second, the prover adds a small amount of binary-symmetric-channel noise R to his PUF measurement in order to improve privacy: the syndrome of X' is slightly changed by this modification, and the communicated error pattern E changes to $E \oplus R$. The noise characteristics of R have to be tuned such that the error correction is not hampered. Note, however, that the drift compensation reduces the amount of noise; hence there is 'room' for R.

A. Proposed Solution

We assume that the prover device has additional non-volatile (NV) memory in which it can store a drift vector $D \in \{0,1\}^n$ and a list $\mathcal{L}$ of up to $N_{\max}$ error patterns observed during the previous executions of the protocol.

The vector D serves to keep track how far the PUF has drifted away from the enrolled PUF measurement X. The reconstruction protocol does error correction with respect to the drifted PUF value X_{drifted}, and then shifts the result by the amount of D. Taking the drifted value X_{drifted} as the zero point for error correction has the advantage that the number of bit errors is reduced. Note that the stored helper data $\tilde{W}$ is the syndrome of X_{drifted}.

System setup:
The same as in Section III-B.

Enrollment:
The same as in Section III-B. In addition, the prover's list $\mathcal{L}$ is initialized to $\emptyset$, and D is initialized to the zero string.

Reconstruction:
1) The prover

   (a) performs a fresh measurement $Y \in \{0,1\}^n$,

(b) adds a small amount of (pseudo)random noise R to Y obtaining $Y' = Y \oplus R$,

(c) computes $\Sigma = \tilde{W} \oplus \mathtt{Syn}(Y')$ and sends Σ to the verifier.

2) The verifier computes the error pattern $E = \mathtt{SynDec}(\Sigma)$ and sends E to the prover.
3) The prover computes $X_{\text{drifted}} = Y' \oplus E$ and the estimators $\hat{X} = X_{\text{drifted}} \oplus D$ and $\hat{K} = \mathtt{KeyDeriv}(\hat{X})$.
4) If $\hat{K} = K$ then the prover performs the following actions.

(a) Add the error pattern $E \oplus R$ to the list $\mathcal{L}$. If necessary, the oldest entry in $\mathcal{L}$ is discarded to make place.

(b) If $\mathcal{L}$ contains $N_{\max}$ entries, check if there are bit positions that occur in most of the entries. If so, construct an error pattern $e \in \{0,1\}^n$ consisting of these positions, replace D by $D \oplus e$, replace the helper data $\tilde{W}$ by $\mathtt{Syn}(X_{\text{drifted}} \oplus e)$, and set $\mathcal{L} = \emptyset$.

B. Security analysis of the proposed protocol

Our solution as proposed in Section VI-A obviously hides the drift D from adversaries who observe E. Thus, the privacy and security problems mentioned in the introduction are mitigated.

Nevertheless, the proposed solution requires storing additional data in the prover's NV memory, namely the drift D and the list $\mathcal{L}$ of error patterns besides the code-offset helper data $\tilde{W}$. In an extended adversary model, one assumes that the adversary has access to all stored data. In this section we study how the security is affected by the additional information revealed to these adversaries.

Theorem 4: An adversary who observes the prover's NV memory has the following amount of uncertainty about X,

$$\begin{aligned}\mathsf{H}(X|\tilde{W}D\mathcal{L}) \;=\; & \mathsf{H}(X|W) - [\mathsf{H}(\tilde{W}|D) - \mathsf{H}(W)] \\ & -I(D;X) - [\mathsf{H}(\mathcal{L}|D\tilde{W}) - \mathsf{H}(\mathcal{L}|X_{\text{drifted}})].\end{aligned} \tag{8}$$

Proof: We write

$$\mathsf{H}(X|\tilde{W}D\mathcal{L}) = \mathsf{H}(X\tilde{W}D\mathcal{L}) - \mathsf{H}(\tilde{W}D\mathcal{L}). \tag{9}$$

Applying the chain rule again we expand these terms as

$$\begin{aligned}\mathsf{H}(X\tilde{W}D\mathcal{L}) &= \mathsf{H}(X) + \mathsf{H}(\tilde{W}D\mathcal{L}|X) \\ &= \mathsf{H}(X|W) + \mathsf{H}(W) \\ &\quad + \mathsf{H}(D|X) + \underbrace{\mathsf{H}(\tilde{W}|DX)}_{0} + \underbrace{\mathsf{H}(\mathcal{L}|\tilde{W}DX)}_{\mathsf{H}(\mathcal{L}|X_{\text{drifted}})}\end{aligned} \tag{10}$$

and

$$\mathsf{H}(\tilde{W}D\mathcal{L}) \;=\; \mathsf{H}(D) + \mathsf{H}(\tilde{W}|D) + \mathsf{H}(\mathcal{L}|D\tilde{W}). \tag{11}$$

In (10) we have used the fact that $\mathcal{L}$ is noise on X_{drifted} and therefore can depend at most on X_{drifted} itself. We substitute (10) and (11) into (9). □

In Theorem 4, the term $\mathsf{H}(X|W)$ is the 'old' result, for the ordinary code offset method. The three other expressions are corrections.

- If all noise is data-independent, then all three correction terms vanish.
- The correction term $\mathsf{H}(\tilde{W}|D) - \mathsf{H}(W)$ is always small.
- If X is uniform and the simple noise model with the parameters p_0, p_1 is adopted, then the correction $I(D;X)$ follows from Theorem 3.
- If the noise is strongly data-dependent, then the correction terms $I(D;X)$ and $\mathsf{H}(\mathcal{L}|D\tilde{W}) - \mathsf{H}(\mathcal{L}|X_{\text{drifted}})$ are not negligible.

Theorem 4 reveals a security issue if a) the noise is data-dependent and b) if the drift significantly breaks the $0 \leftrightarrow 1$ symmetry. The new part of the helper data – drift D and List $\mathcal{L}$ – reveals more information about the PUF key (via leakage about X) then the old part of W. Thus, in the worst case scenario with a data-dependent noise and an asymmetric drift our proposed protocol shifts the vulnerable part of the old Reverse FE procotol (the error pattern) to the device non-volatile memory (the Helper Data).

VII. Conclusion

We addressed leakage issues of the Reverse Fuzzy Extractor. In particular, we argued that a) a security risk exists if the PUF bias drift is not independent of the PUF biases themselves, and b) the drift causes a privacy risk. We modeled the drift and evaluated our model on real data, confirming the existence of the drift. We analyzed the leakage in a simplified model. We proposed a modified Reverse FE protocol that solves most of the leakage issues. In the case of eavesdropping adversaries, the privacy leakage and key leakage are completely eliminated. In the case of adversaries who have access to the enrollment data, our solution's security is specified by the correction terms in Theorem 4, which can be dangerous only if the noise is strongly data-dependent. In future work we will test our improved Fuzzy Extractor on real data to get detailed insights in its resilience.

References

[1] P. Tuyls, G.-J. Schrijen, F. Willems, T. Ignatenko, and B. Škorić, "Secure key storage with PUFs," *Security with Noisy Data-On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, pp. 269–292, 2007.

[2] U. Rührmair, H. Busch, and S. Katzenbeisser, "Strong PUFs: Models, Constructions, and Security Proofs," in *Towards Hardware-Intrinsic Security*, 2010, pp. 79–96.

[3] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Short Paper: Lightweight Remote Attestation Using Physical Functions," in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec '11, 2011, pp. 109–114.

[4] A. Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, 2012, vol. 7397, pp. 374–389.

[5] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," in *Towards Hardware-Intrinsic Security*, ser. Information Security and Cryptography, 2010, pp. 3–37.

[6] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[7] B. Škorić and N. de Vreede, "The Spammed Code Offset Method," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 5, pp. 875–884, May 2014.

[8] R. van den Berg, B. Škorić, and V. van der Leest, "Bias-based Modeling and Entropy Analysis of PUFs," in *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices*, ser. TrustED '13, 2013, pp. 13–20.