Eliminating Leakage in Reverse Fuzzy Extractors

A. Schaller, T. Stanko, B. Škorić, S. Katzenbeisser

***Abstract*—In recent years Physically Unclonable Functions (PUFs) have been proposed as a promising building block for key storage and device 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 (due to a systematic drift of the PUF responses, the error pattern is linkable to the identity) and even security problems (if the noise is data-dependent). In this work we investigate both these issues and propose modified protocols that eliminate the problems.**

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 can be applied to various identification and authentication applications. Several protocols featuring PUFs have been devised in the past, 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 this work, we will consider the use of a PUF-based key storage in the context of privacy-preserving protocols that are designed to hide the identity of the users 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 removed before the measurement can be used 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 this redundancy data is public and thus can be accessed by 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. 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. In [4] it was argued that the PUF key is secure as long as the measurement noise is independent of the PUF value itself.

In this paper we (a) examine what happens when this assumption does not hold, i.e., we study the security implications of data-dependent noise; and (b) argue that the statement "the PUF key is secure as long as the measurement noise is data-independent", while true, does not cover all security aspects of the protocol.

Point (a) is important because data-dependent noise was shown to exist in PUFs such as FlipFlop PUFs, Latch PUFs and Buskeeper PUFs [5]. We use the Binary Asymmetric Channel (BAC) as our noise model. We quantify the leakage in this model and study possible countermeasures. It turns out that applying an extra Z-channel [6] after the BAC is a very effective solution.

Regarding point (b), we note that PUFs exhibit a slow 'drift' in the values of their response bits over time, which is due to device ageing. This drift is characteristic to individual PUF instances. An attacker can try to identify PUF instances by analyzing the revealed error pattern, since the drift is directly reflected in the error pattern. This creates a *privacy* risk in case the PUF is used in privacy-preserving protocols. It is worth noting that the privacy leakage can increase the above-mentioned security risk by allowing the attacker to link protocol executions from the same PUF. We show from experimental data that several PUF types indeed exhibit a drift. We propose an adaptation of the Reverse FE protocol that eliminates the drift issue.

The rest of the paper is organized as follows. In Section II we define notations, give a brief overview on PUFs and Fuzzy Extractors, introducing the Reverse Fuzzy Extractor in partiuclar. In Section III we discuss the problem of data-dependent noise and describe our solution. In Section IV we look at experimental data on drift and analyse the leakage caused by drift. We introduce an improved version of the Reverse Fuzzy Extractor in Section V.

II. Preliminaries

A. Notation and terminology

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

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

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

The Binary Asymmetric Channel (BAC) is a memory-less channel. An transmitted bit X is received as a noisy bit X'. The channel is fully characterized by two parameters: $\alpha \overset{\text{def}}{=} \Pr[X'=1|X=0]$ and $\beta \overset{\text{def}}{=} \Pr[X'=0|X=1]$. Without loss of generality we consider only $\alpha, \beta \in [0, \frac{1}{2}]$. The case $\alpha = \beta$ is called the Binary Symmetric Channel (BSC). The case $\alpha = 0$ or $\beta = 0$ is known as a Z-channel. We will occasionally write $\alpha = \mu - \delta$, $\beta = \mu + \delta$, with $\mu \in [0, \frac{1}{2}]$ and $|\delta| \leq \min(\mu, \frac{1}{2} - \mu)$.

B. 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 [7]. 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 '0' or '1'. 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 be based on random timing characteristics of circuits, among them Ring Oscillator PUFs and Arbiter PUFs.

Due to physical characteristics of the device, measurements of a PUF response are subject to noise; thus, subsequent measurements will be slightly different. In order to use them in cryptographic protocols, the noisy responses must be stabilized. This is done by employing a Fuzzy Extractor [8], which extracts the stable part of the PUF response and transforms it to a uniformly distributed value.

C. Fuzzy Extractors

The authors of [8] 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.

D. The Reverse Fuzzy Extractor

We briefly review the Reverse FE 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 [8], [9] with the sole difference that syndrome decoding is outsourced to the verifier.

[1]We will actually work with a more general primitive: a Secure Sketch. It is always possible to construct a Fuzzy Extractor from a Secure Sketch.

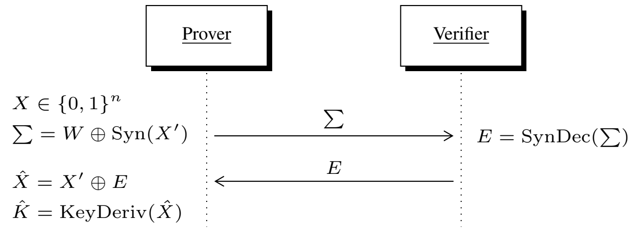

Prover
Verifier
$X \in \{0,1\}^n$
$\sum = W \oplus \mathrm{Syn}(X')$
$\sum$
$E = \mathrm{SynDec}(\sum)$
E
$\hat{X} = X' \oplus E$
$\hat{K} = \mathrm{KeyDeriv}(\hat{X})$

Fig. 1. Sequence diagram of the device authentication protocol.

System setup:
The parties agree on a linear error correcting code $\mathcal{C}$, with message length k and codeword length n. The encoding algorithm of $\mathcal{C}$ 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:
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. The verifier computes the error pattern $E = \texttt{SynDec}\, \Sigma$ and sends E to the prover. 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.

III. Data-dependent noise

A. Quantifying the problem

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 (Section II-D). 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 '0' in X is much more likely than a '1'. This is a *security* risk. It becomes even more serious if the adversary observes multiple transcripts from the same prover, carrying different information about X, and is able to link those transcripts together.

We adopt the Binary Asymmetric Channel (see Section II-A) as our noise model and quantify the amount of leakage in this model. We further assume, for simplicity, that the bias is constant over the device, i.e., we consider a global bias.

Lemma 1: Let $X \in \{0,1\}^n$ be the enrollment measurement, with i.i.d. bits $X_i \sim (1-p,p)$, i.e., all bits have the same bias $\Pr[X_i=1]=p$. Let $X' \in \{0,1\}^n$ be the reconstruction measurement. Let the noise behave as a BAC. Let $E = X \oplus X'$ be the error pattern during reconstruction. Then the mutual information between the error pattern and X is given by

$$I(X;E) = n\Big[h\Big((1-p)\alpha + p\beta\Big) - (1-p)h(\alpha) - ph(\beta)\Big] \quad (2)$$

and the entropy of X given E is

$$\mathsf{H}(X|E) = nh(p) - I(X;E). \quad (3)$$

The proof is given in Appendix A. This shows that if $\alpha=\beta$, then $I(X;E)=0$ and no leakage occurs.

More generally, an attacker could observe multiple error vectors $E^{(1)},\ldots,E^{(k)} \in \{0,1\}^n$ from the same device. We define T_i as the number of observations that yield an error in location i, i.e., $T_i = |\{k:\ E_i^{(k)}=1\}| = \sum_{j=1}^{k} E_i^{(j)}$. We define $T=(T_i)_{i=1}^n$. The generalization of (2) then becomes

$$\begin{aligned} I(X;E^{(1)}\cdots E^{(k)}) &= I(X;T) \\ &= \mathsf{H}(T) - \mathsf{H}(T|X) \\ &= n\mathsf{H}(T_i) - n\mathsf{H}(T_i|X_i), \end{aligned} \quad (4)$$

where in the last line the index i is arbitrary. For the evaluation of $\mathsf{H}(T_i|X_i)$ and $\mathsf{H}(T_i)$ we need the corresponding probability distributions. For given X_i, the T_i is binomial-distributed: $\Pr[T_i=t|X_i=0] = \binom{k}{t}\alpha^t(1-\alpha)^{k-t}$ and $\Pr[T_i=t|X_i=1] = \binom{k}{t}\beta^t(1-\beta)^{k-t}$. This yields $\Pr[T_i=t] = (1-p)\Pr[T_i=t|X_i=0] + p\Pr[T_i=t|X_i=1]$.

Fig. 2 shows the leakage $nI(X_i;T_i)$ relative to the total information $n\mathsf{H}(X_i)=nh(p)$ that can potentially be leaked, i.e., the leaked fraction. This is shown for various parameter settings of μ and k, where p has been tuned so as to maximize $\mathsf{H}(X_i|T_i)$, the attacker's uncertainty about X_i. While k is the number of observed error instances observed by the attacker, μ indicates the asymmetry of the BAC (see Section II-A). The leakage is considerable. For example, in the $\mu=0.05$ graph we see that already at $\delta=0.025$ a dozen observations reveals around 10% of the entropy of X. In order to connect Fig. 2 to real-life PUFs we cite some numbers from the Unique dataset [10]. For Ring Oscillator PUFs, DFF PUFs (high temperature) and Latch PUFs (low temperature), values of δ up to 0.07, 0.16 and 0.19 respectively can occur (at $\mu \approx 0.2$), while Arbiter and SRAM PUFs have very little asymmetry. These are the 'raw' values before reliable component selection has been applied, or other processing, e.g. repetition codes, that reduces μ and δ. It is clear from Fig. 2 that even after noise reduction residual asymmetries lead to significant leakage.

A naive attempt to deal with the leakage problem would be to tune the `KeyDeriv` function so that it compresses X more strongly, taking into account the expected leakage; however, there is no clear upper bound on the leakage, as the adversary can eavesdrop on additional protocol rounds. Keeping in mind that even a few percent of key leakage can endanger the cryptographic primitives, we conclude that, no matter how

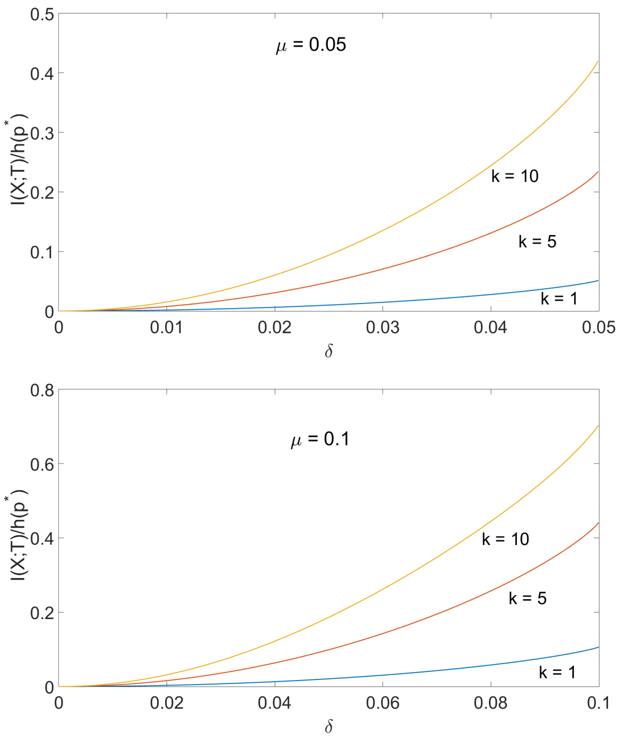


Fig. 2. *The attacker's knowledge about X_i after observing k error instances, for various BAC parameters and various k. In each plotted point individually, the p is tuned to maximize $\mathsf{H}(X_i|T_i)$ as a function of μ, δ and k.*

`KeyDeriv` and the distribution of X are tuned, the Reverse FE has a serious leakage problem when the noise is data-dependent.

B. Eliminating the leakage

In order to eliminate the leakage, we propose a simple solution in the BAC case: to apply, in the reconstruction phase, an additional Z-channel that compensates the asymmetry in the $X \to X'$ channel. The parameters required for the Z-channel can be pre-computed based on calibration measurements which are done at system setup or at enrollment. The adapted reconstruction procedure is as follows.

Reconstruction:

1) The prover performs a fresh measurement $X' \in \{0,1\}^n$. He applies additional Z-channel noise to X', yielding X''. He computes $\Sigma = W \oplus \mathtt{Syn}(X'')$ 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 the estimators $\hat{X} = X'' \oplus E$ and $\hat{K} = \mathtt{KeyDeriv}(\hat{X})$.

We define the notation $\alpha_\mathrm{z} = \Pr[X_i''=1|X_i'=0]$ and $\beta_\mathrm{z} = \Pr[X_i''=0|X_i'=1]$ for the Z-channel bit flip probabilities (see Fig. 3). Note that at least one of the parameters $\alpha_\mathrm{z}, \beta_\mathrm{z}$ is zero. The nonzero parameter is tuned such that the combined channel, consisting of the BAC with appended Z-channel, is a BSC. We will denote the bit error rate of this BSC as ε. The parameter tuning is given by the following theorem.

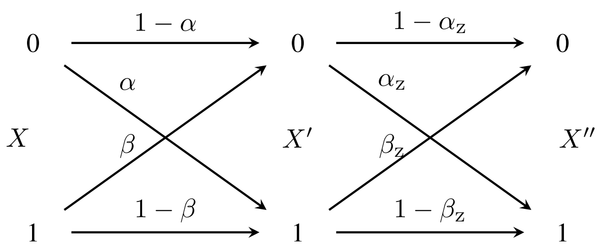


Fig. 3. *Concatenation of two Binary Asymmetric Channels.*

Theorem 2: Let $X \to X'$ be a given BAC with parameters α, β (or μ, δ). Let $X' \to X''$ be a second BAC with parameters α_Z, β_Z such that the combined channel $X \to X''$ is a BSC with bit error rate ε. Then ε is minimised by the following parameter choice:

$$\alpha \geq \beta\ (\delta \leq 0): \qquad \alpha_Z = 0,\ \beta_Z = \frac{\alpha-\beta}{1+\alpha-\beta} = \frac{2|\delta|}{1+2|\delta|}$$

$$\beta \geq \alpha\ (\delta \geq 0): \qquad \beta_Z = 0,\ \alpha_Z = \frac{\beta-\alpha}{1+\beta-\alpha} = \frac{2\delta}{1+2\delta}.$$

Both cases yield $\varepsilon = \frac{\mu+|\delta|}{1+2|\delta|}$.

The proof is given in Appendix B. (Note that Theorem 2 does not assume that a Z-channel is the solution but starts more generically from a second BAC.)

Our solution entirely eliminates leakage from the communicated error pattern, but this comes at a cost: the additional noise degrades the channel, i.e. it reduces the amount of useful information about X that can be recovered after error correction. We now quantify how much 'worse' the channel $X \to X''$ is than the original channel $X \to X'$. First we show that the final noise parameter ε cannot be larger than the highest BAC parameter.

Corollary 3: The bit-error probability ε specified in Theorem 2 satisfies

$$\varepsilon \in \Big[\frac{\alpha+\beta}{2}, \max\{\alpha,\beta\}\Big] = [\mu, \mu+|\delta|]. \tag{5}$$

Proof: Let $\beta \geq \alpha$ ($\delta \geq 0$) w.l.o.g. From Theorem 2 we have $\varepsilon = \frac{\mu+\delta}{1+2\delta}$. Obviously, $\varepsilon \leq \mu+\delta$. Furthermore, $\varepsilon = \frac{\mu+\delta}{1+2\delta} = \frac{\mu(1+2\delta)+\delta-2\delta\mu}{1+2\delta} = \mu + \frac{\delta(1-2\mu)}{1+2\delta} \geq \mu$. In the last step we used $0 \leq \mu \leq \frac{1}{2}$. Thus, we obtain $\varepsilon \in [\mu, \mu+\delta] = \big[\frac{\alpha+\beta}{2}, \beta\big]$. The derivation for $\delta < 0$ follows exactly the same lines. ∎

Next we characterize the loss of channel quality by looking at the *channel capacity*. The channel capacity places a lower bound on how much source entropy (from X) is required to derive a noise-robust key of a certain size. A capacity equal to 1 corresponds to absence of noise, in which case all entropy from X is directly usable. In general, the capacity C is the *fraction* of all the entropy in X that may survive error correction in case of an ideal error-correcting code.

The BSC $X \to X''$ has capacity $C_{\rm BSC} = 1 - h(\varepsilon)$, with $\varepsilon = \frac{\mu+|\delta|}{1+2|\delta|}$ as specified in Theorem 2. The capacity of the BAC is given by (see e.g. [11]),

$$\begin{aligned} C_{\rm BAC} &= \frac{\mu-|\delta|}{1-2\mu} h(\mu+|\delta|) - \frac{1-\mu-|\delta|}{1-2\mu} h(\mu-|\delta|) \\ &\quad + \log\Big(1 + 2^{-\frac{h(\mu+|\delta|)-h(\mu-|\delta|)}{1-2\mu}}\Big) \qquad (6) \\ &= 1 - h(\mu) + (\frac{\delta}{\mu})^2 \frac{\mu}{2\ln 2} + \mathcal{O}(\frac{\delta^2}{\mu^2}[\mu \ln \mu]^2). \qquad (7) \end{aligned}$$

Theorem 4: The capacity loss due to introducing the Z-channel can be approximated as

$$\begin{aligned} C_{\rm BAC} - C_{\rm BSC} &= |\delta|(1-2\mu)\log\frac{1-\mu}{\mu} + \mathcal{O}(\delta^2) \\ &= |\delta| \log\frac{1}{\mu} + \mathcal{O}(\delta\mu\log\frac{1}{\mu}). \qquad (8) \end{aligned}$$

Proof: Follows from Taylor-expanding the expressions for $C_{\rm BAC}$ and C_{BSC}. ∎

In Fig. 4 we plot the capacity loss $C_{\rm BAC} - C_{\rm BSC}$ relative to the original capacity $C_{\rm BAC}$. The 'raw' noise levels in PUFs (i.e. without reliable cell selection), for instance for D-Flipflop PUFs as reported in [5], can be as high as $\mu = 0.2, \delta = 0.1$. According to Fig. 4 the Z-channel insertion would then lead to almost 40% capacity loss. Depending on the context this may be acceptable. If not, the noise μ, δ can be reduced by techniques such as reliable component selection and repetition codes. In Fig. 4 we see that the capacity loss is less severe at low noise. The optimal tuning of the parameters in the noise reduction techniques depends on the specific PUF properties.

Of course the legitimate parties need to estimate the noise parameters α, β of the BAC in order to be able to set α_Z, β_Z as specified in Theorem 2. The noise parameters have to be established either (i) as part of a system setup phase before the enrollments or (ii) during operation, by using a subset of the PUF cells as non-secret test cells for calibration purposes.

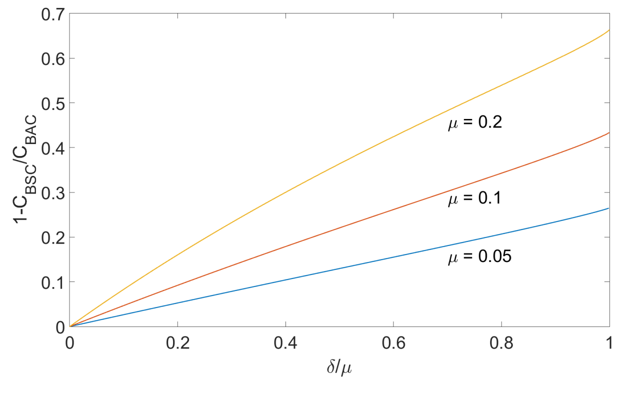


Fig. 4. Relative loss of channel capacity due to the extra Z-channel.

IV. The drift problem

In some PUF instances individual cells have a bias towards either zero or one. We present measurements which show that these biases change over time; we call this the *drift* of a PUF. Furthermore, we provide a model for the drift and estimate the privacy leakage (and the induced key leakage) in Reverse Fuzzy Extractors due to the drift. The following evaluation of the drift and its results have been presented in a previous paper [12].

A. Drift model

We adopt the bias-based PUF model proposed in [5]. We denote the bias of a PUF cell 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' response during k enrollment measurements: $\hat{b}_i = x_i/k \in \{0, \frac{1}{k}, \ldots, 1\}$. A PUF is fully characterized by a vector of biases, $\boldsymbol{b} = (b_i)_{i=1}^n$. Similarly, b'_i represents the bias of cell i at a later time. It can be estimated from the number x'_i of '1' responses in a series of l PUF responses: $\hat{b}'_i = x'_i/l \in \{0, \frac{1}{l}, \ldots, 1\}$.

The drift is modeled by a transition probability $\tau(\boldsymbol{b}'|\boldsymbol{b})$ indicating how likely it is that the PUF has bias vector $\boldsymbol{b}'$ at a later time given that it had $\boldsymbol{b}$ at enrollment. Assuming that the n PUF cell responses are mutually independent (this assumption seems justified as we did not see any correlation between cell responses 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

$$\tau(\boldsymbol{b}'|\boldsymbol{b}) = \prod_{i\in[n]} \tau_0(b'_i|b_i). \tag{9}$$

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

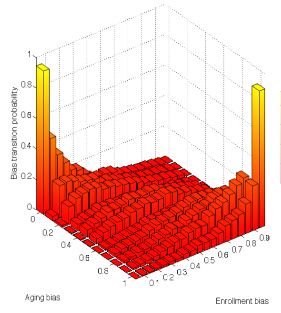
B. Drift data

We made use of PUF measurement data obtained in the UNIQUE project[2]. In this project custom ASICS with different PUF types, including SRAM, latch, D-Flip-Flop (DFF), Arbiter and Ring Oscillator (RO) PUFs, were developed and tested under different conditions. The UNIQUE data set includes measurements of PUFs 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 device can be simulated in short time.

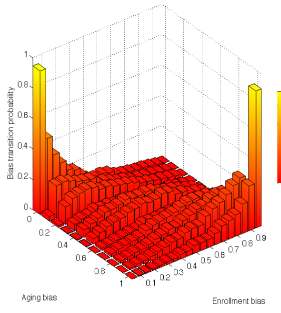
Three different datasets were available for our experiments: enrollment data taken right after manufacturing (referred to as time t_0), measurements at the beginning of the aging process (at time t_1) and measurements after the aging process had terminated (time t_2). Measurements at t_1 correspond to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

Figure 5 shows τ_0 for SRAM, latch, DFF and RO PUFs, for $t_0 \to t_1$ and $t_0 \to t_2$ aging. In 5a, 5c and 5e we observe a diagonal 'saddle' between $(0,0)$ and $(1,1)$ for the $t_0 \to t_1$ data. This indicates that SRAM, latch and DFF PUFs have a stable bias over a short operating time. The RO PUF (5g) is

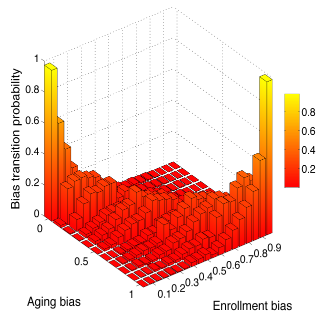
[2]http://www.unique-project.eu



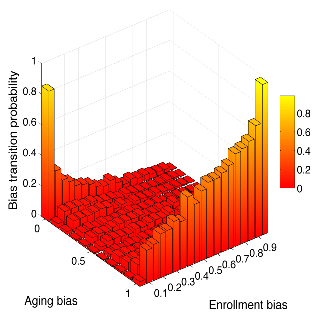
(a) SRAM: $t_0 \to t_1$



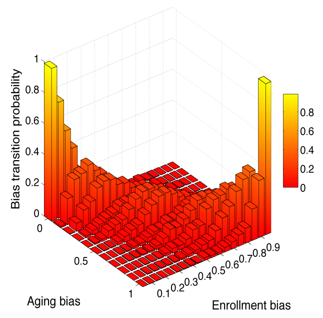
(b) SRAM: $t_0 \to t_2$



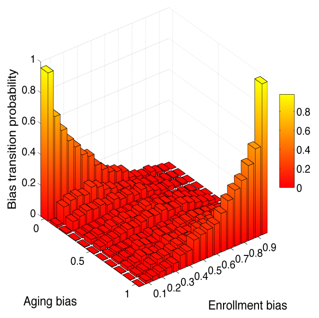
(c) Latch: $t_0 \to t_1$



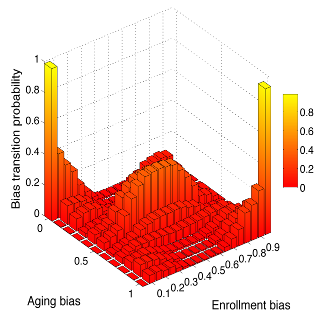
(d) Latch: $t_0 \to t_2$



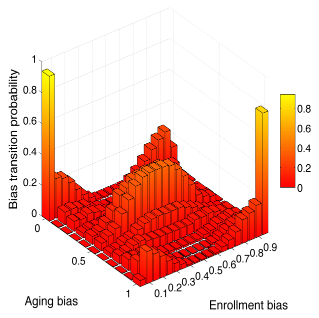
(e) DFF: $t_0 \to t_1$



(f) DFF: $t_0 \to t_2$



(g) RO: $t_0 \to t_1$



(h) RO: $t_0 \to t_2$

Fig. 5. *Bias transition probabilities $\tau_0(b'|b)$ for SRAM, latch, DFF and RO PUFs.*

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 not unexpected, as ring oscillators can be used to generate random numbers as well.

At t_2 we see a flattening of the 'saddle' for all PUF types (5b, 5d, 5f, 5h). This indicates, as expected, that there is a significant drift after a longer operation time. Note that not all

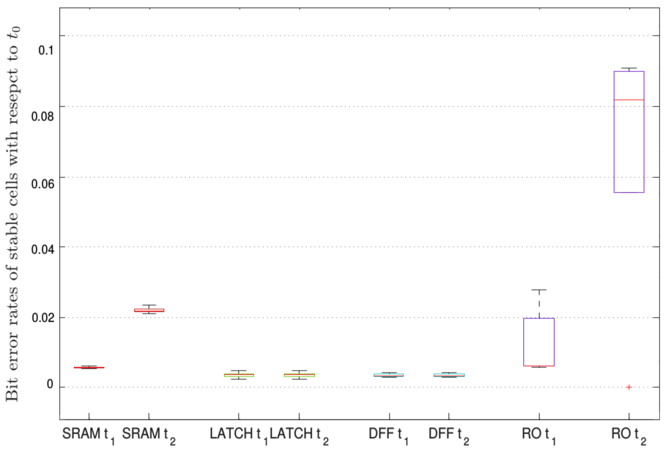


Fig. 6. *Bit error rates of stable cells for various PUF types after $t_0 \to t_1$ and $t_0 \to t_2$ aging. The red line in each box indicates the median. The colored bottom and top of each box marks the 25th/75th percentile. The height of a box displays the inter quartile range (IQR). The whisker's ends indicate the lowest and highest bit error rates that are within 1.5 times the IQR. Single outsider values are marked by red plus signs.*

the transition probabilities are symmetric under $0 \leftrightarrow 1$ reversal; this phenomenon mainly occurs 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 during reconstruction. Instead, fine-grained bias information is used only for the selection of reliable cells. A FE will typically store pointers to stable cells (i.e., cells that have an enrollment bias close to 0 or 1); only those are then used for key derivation.

For this context we introduce a simplified drift model in which the biases are binarized to 0/1 values, and only reliable cells are taken into account. The model has only two parameters: α_d, the probability of a $0 \to 1$ bit transition due to drift, and β_d, the probability of a $1 \to 0$ transition due to drift. The numerical values of these parameters slowly vary as a function of time. Table I lists transition probabilities of 'stable' cells with enrollment biases $b_i \in [0, 0.05] \cup [0.95, 1]$. Fig. 6 shows the same data graphically. For SRAM and RO PUFs, the $t_0 \to t_2$ bit error rate is considerably higher than the $t_0 \to t_1$ bit error rate.

C. Leakage analysis

The results of Section IV-B 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 $E = D \oplus N$. This both has an impact on security and privacy.

Privacy

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

Lemma 5: 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 α_d, β_d as defined above. Then the Hamming distance between D_1 and D_2 is binomial-distributed, with parameters n and P_uneq, where

$$P_\mathrm{uneq} = 2\frac{\alpha_\mathrm{d} + \beta_\mathrm{d}}{2}(1 - \frac{\alpha_\mathrm{d} + \beta_\mathrm{d}}{2}). \tag{10}$$

The proof is given in Appendix C.

Corollary 6: 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 α_d, β_d. Then the expected Hamming distance between D_1 and D_2 is $\mu_\mathrm{HD} = nP_\mathrm{uneq}$, and the variance is $\sigma^2_\mathrm{HD} = nP_\mathrm{uneq}(1 - P_\mathrm{uneq})$.

Proof: Follows from Lemma 5 and the properties of the binomial distribution. ■

If $\alpha_\mathrm{d} + \beta_\mathrm{d}$ is large enough, and if the short-timescale noise N does not mask the drift, then the observed noise pattern E, via the constant part D, becomes a unique characterizing property for each PUF, as quantified in Corollary 6.

For the further analysis we introduce the following notation. We denote the set of observed error patterns as $\mathcal{E} = (E_a)_{a=1}^{\ell}$, where ℓ is the number of observations. Similarly we define $\mathcal{N} = (N_a)_{a=1}^{\ell}$, with $E_a = D \oplus N_a$. We write $X_\mathrm{drifted} = X \oplus D$.

Theorem 7: Let $\mathcal{N}_\mathrm{av} \in \{0,1\}^n$ be the pattern obtained by averaging $\mathcal{N}$. The amount of information about D gained from observing $\mathcal{E}$ is given by

$$I(\mathcal{E}; D) = \mathsf{H}(D \oplus \mathcal{N}_\mathrm{av}) - \mathsf{H}(\mathcal{N}_\mathrm{av}). \tag{11}$$

Proof: see Appendix D.

If the noise N_a is data-independent, then the adversary can get a good estimate of D by averaging the error patterns, and we can almost say that observing $\mathcal{E}$ is the same as observing D and $\mathcal{E}$ (or, equivalently, D and $\mathcal{N}$). In the case of data-dependent noise N_a the situation becomes more complicated. In terms of Markov chains we have $X \to D$ and $X_\mathrm{drifted} \to N_a$. Hence, a data-dependent short-timescale noise N_a has some weak dependence on the long-timescale drift D via $X_\mathrm{drifted} = X \oplus D$.

Security

Next we analyze the security implications if the adversary is able to link multiple instances of the authentication protocol run by the same PUF device. (Either because of the above explained privacy problem or by some other means.)

Since we did not specify the `KeyDeriv` algorithm, we cannot compute the mutual entropy between $\mathcal{E}$ and the PUF key K in general. Instead, we derive a bound on the mutual information between $\mathcal{E}$ and X.

Theorem 8: The leakage about X caused by observation of the error patterns $\mathcal{E}$ can be upper bounded as

$$I(\mathcal{E}; X) \le I(D; X) + I(\mathcal{N}; X_\mathrm{drifted}), \tag{12}$$

Proof: see Appendix E.

TABLE I. TRANSITION PROBABILITIES $0 \to 1$ AND $1 \to 0$ FOR $t_0 \to t_1$ AND $t_0 \to t_2$ AGING (BIASES $b_i \in [0, 0.05] \cup [0.95, 1]$. MEAN$\pm$STANDARD DEVIATION IS LISTED).

Period	Transition	SRAM	LATCH	DFF	RO
$t_0 \to t_1$	$0 \to 1$	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
$t_0 \to t_2$	$0 \to 1$	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

The two leakage terms in Theorem 8 are very similar. The $I(\mathcal{N}; X_{\rm drifted})$ term is exactly the leakage shown in Fig. 2, but now about $X_{\rm drifted}$ instead of X, which is practically the same from a security point of view. The mutual information $I(D; X)$ is precisely given by Lemma 1 where the error pattern E is replaced by the drift D, and the parameters α, β by $\alpha_{\rm d}, \beta_{\rm d}$. The $I(D; X)$ is nonzero if the drift is asymmetric.

Note that, in contrast to the leakage $I(\mathcal{N}; X_{\rm drifted})$, the existence of the $I(D; X)$ leakage does not necessarily imply that there is a grave security problem: The drift D is a single error pattern, whereas measurements of short-term asymmetric noise reveal new information every time. A properly designed extraction procedure `KeyDeriv` can compensate for the leakage $I(D; X)$ by sufficiently compressing X. In case privacy is not important, we see the leakage $I(D; X)$ primarily as an issue that reduces the efficiency of the Fuzzy Extractor.

Finally we briefly comment on the case where the adversary observes the helper data W as well as the communicated noise patterns $\mathcal{E}$.

Theorem 9: The leakage caused by observing W and $\mathcal{E}$ can be bounded as

$$I(W\mathcal{E}; X) \leq I(W; X) + I(\mathcal{E}; X). \tag{13}$$

Proof: See Appendix F.

The bound in Theorem 9 is tight, since $\mathsf{H}(\mathcal{E}|W) \approx \mathsf{H}(\mathcal{E})$. Thus we can also read Theorem 9 as $I(W\mathcal{E}; X) \approx I(W; X) + I(\mathcal{E}; X)$, i.e. leakage from W plus *almost independent* leakage from $\mathcal{E}$.

V. SOLVING THE DRIFT PROBLEM

In this section we present a modified Reverse Fuzzy Extractor in which the protocol messages do not cause leakage, even if there is PUF drift. 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. This technique is compatible with the addition of a Z-channel as described in Section III-B.

A. Proposed solution for the drift problem

In a nutshell our proposal is as follows. The prover device has additional non-volatile (NV) memory in which it stores an estimated drift vector $\hat{D} \in \{0,1\}^n$ and a list $\mathcal{L}$ of up to $N_{\max}$ error patterns observed during previous executions of the protocol. The $\hat{D}$ serves to keep track of how far the PUF has drifted away from the enrolled PUF measurement X. The reconstruction protocol does error correction with respect to the (estimated) drifted PUF value $\hat{X}_{\rm drifted}$, and then shifts the result by the amount of $\hat{D}$. Taking the drifted value $\hat{X}_{\rm drifted}$ as the zero point for error correction has the additional advantage that the number of bit errors is reduced. The stored helper data is always equal to $\tilde{W} = \mathtt{Syn}\, \hat{X}_{\rm drifted}$.
A more detailed description of our proposal is given below.

System setup:
The same as in Section II-D.

Enrollment:
The same as in Section II-D. The enrolled helper data is $\tilde{W} = \mathtt{Syn}\, X$. In addition, the prover's list $\mathcal{L}$ is initialized to the empty string $\emptyset$, and $\hat{D}$ is initialized to the zero string.

Reconstruction:
1) The prover
   (a) performs a fresh measurement $Y \in \{0,1\}^n$,
   (b) adds (pseudo)random Z-channel noise R, yielding $Y' = Y \oplus R$.
   (c) computes $\Sigma = \tilde{W} \oplus \mathtt{Syn}(Y')$ and sends Σ to the verifier.
2) The verifier computes the error pattern $\tilde{E} = \mathtt{SynDec}(\Sigma)$ and sends $\tilde{E}$ to the prover.
3) The prover computes $\hat{X}_{\rm drifted} = Y' \oplus \tilde{E}$ and the estimators $\hat{X} = \hat{X}_{\rm drifted} \oplus \hat{D}$ and $\hat{K} = \mathtt{KeyDeriv}(\hat{X})$.
4) If $\hat{K} = K$ then the prover performs the following actions.
   (a) Add the error pattern $\tilde{E} \oplus R$ 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 are '1' in most of the entries. If so, construct an error pattern $e \in \{0,1\}^n$ consisting of these positions, replace $\hat{D}$ by $\hat{D} \oplus e$, and replace the helper data $\tilde{W}$ by $\mathtt{Syn}(\hat{X}_{\rm drifted} \oplus e)$. Xor all entries in $\mathcal{L}$ with e.

B. Privacy of the proposed protocol

We have $Y' = X \oplus D \oplus N \oplus R$, where N is short-timescale BAC noise, and (in case of correct reconstruction of X) we have $\hat{X}_{\rm drifted} = X \oplus \hat{D}$. This gives

$$\tilde{E} = Y' \oplus \hat{X}_{\rm drifted} = (D \oplus \hat{D}) \oplus N \oplus R. \tag{14}$$

Thus, the error pattern $\tilde{E}$ observed by the adversary is a combination of (i) Z-channel-compensated (and hence symmetric) short-timescale noise $N \oplus R$, and (ii) a small long-timescale component $D \oplus \hat{D}$ which vanishes if the estimator $\hat{D}$ is accurate.

Given accurate $\hat{D}$, there is no long-timescale structure to be observed in $\tilde{E}$. Furthermore, the symmetry of the noise $N \oplus R$ (as opposed to N) guarantees that the adversary learns

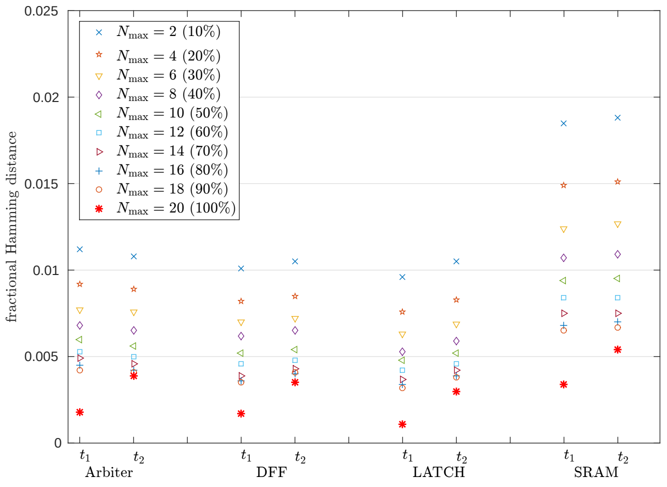


Fig. 7. *Accuracy of the approximated drift vector $\hat{D}$ compared to the actual drift vector D, given as fractional Hamming distances. The actual drift D is based on 20 reconstruction measurements, taken at time periods t_1 & t_2. Fractional Hamming distances are given as mean values over 40 PUF instances.*

nothing about the data X_{drifted}. Thus, both privacy aspects are solved.

We checked the accuracy of $\hat{D}$ compared to D, by simulating the proposed protocol based on the same data that was used for evaluating the systematic drift in Section IV-B. Figure 7 shows the fractional Hamming distance between $\hat{D}$ and D as a function of $N_{\max}/20$ for various PUF types at time periods t_1 and t_2. The listed values are averages of 40 individual PUF instances. As expected, with increasing $N_{\max}$, the accuracy of $\hat{D}$ improves up to the point where the entire data set is considered ($N_{\max} = 20$), resulting in $\hat{D} \approx D$, i.e., a fractional Hamming distance close to zero. (Not exactly zero due to quantisation noise.) The results show that $\hat{D}$ deviates only by 2% from the actual drift in the worst case (considering only two measurements) and thus demonstrate that the approximated drift $\hat{D}$ is accurate enough in order to mask the long-timescale drift.

C. Security of the proposed protocol

As mentioned above, an adversary who observes $\tilde{E}$ learns nothing about X_{drifted}; this is the case because of the Z-channel insertion just as in Section III-B. However, in the new protocol we have to store additional public data $\hat{D}, \mathcal{L}$ together with $\tilde{W}$ in the NV memory. We study how the security is affected by this additional information.

Theorem 10: An adversary who observes the prover's NV memory has the following amount of information about X,

$$I(X; \tilde{W}\hat{D}\mathcal{L}) = I(X;W) + [\mathsf{H}(\tilde{W}|\hat{D}) - \mathsf{H}(W)] + I(\hat{D};X) + [\mathsf{H}(\mathcal{L}|\hat{D}\tilde{W}) - \mathsf{H}(\mathcal{L}|X_{\text{drifted}})]. \quad (15)$$

In Theorem 10, the term $I(X;W)$ is the 'old' result, for the ordinary code offset method. The corresponding proof can be found in the Appendix.

- The $I(\hat{D};X)$ is nonzero if the drift is asymmetric. As mentioned in Section IV-C, a nonzero leakage here is not a severe problem and can be dealt with by properly choosing the parameters in the function `KeyDeriv`.
- The term $\mathsf{H}(\tilde{W}|\hat{D}) - \mathsf{H}(W)$ is small, since $\tilde{W}$ equals the original helper data W compensated by the drift.
- The term $\mathsf{H}(\mathcal{L}|\hat{D}\tilde{W}) - \mathsf{H}(\mathcal{L}|X_{\text{drifted}})$ is negligible, since the noise stored in $\mathcal{L}$ is symmetric and hence data-independent.

Note that our scheme has moved the leakage term $I(X;\hat{D})$ from the eavesdropping domain to the NV memory domain.

VI. Conclusion

We addressed two leakage issues of the Reverse Fuzzy Extractor protocol. In particular (a) data-dependent short-timescale noise poses a severe security problem; (b) there is privacy-sensitive leakage if the long-timescale PUF drift depends on the PUF biases themselves.

Our study of experimental data confirms the existence of asymmetric (data-dependent) drift in several types of PUF.

We propose two modifications to the Reverse FE scheme which together eliminate both leakage problems, (i) additional Z-channel noise that turns a BAC into a BSC; (ii) drift compensation by storing the estimated drift and recent error patterns in the prover device.

The first modification turns the noisy channel $X \to X'$ into an even more noisy channel $X \to X''$. Fig. 4 summarises the capacity loss, which seems acceptable in practical cases.

The second modification 'moves' the drift problem from the eavesdropping domain to the helper data domain: In the new scheme, an adversary who gets access to the data stored in the NV memory of the prover device will learn as much as an eavesdropper did in the original reverse FE scheme. On the other hand, an attacker who only eavesdrops will not be able to identify the PUF device.

Acknowledgements

This work has been supported by the European Commission through the ICT program under contract INFSO-ICT-284833 (PUFFIN) and by the Netherlands Organisation for Scientific Research NWO through Cyber Security project 628.001.019 (ESPRESSO).

Appendix

A. Proof of Lemma 1

Since all bits are independent we consider a single bit i. We have $\Pr[E_i = 1] = \Pr[E_i = 1|X_i = 0]\Pr[X_i = 0] + \Pr[E_i = 1|X_i = 1]\Pr[X_i = 1] = (1-p)\alpha + p\beta$. Thus, $\mathsf{H}(E_i) = h\Big((1-p)\alpha + p\beta\Big)$. Next we have $\mathsf{H}(E_i|X_i) = \Pr[X_i = 0]h(\alpha) + \Pr[X_i = 1]h(\beta)$. We obtain $I(X_i;E_i) = \mathsf{H}(E_i) - \mathsf{H}(X_i|E_i) = h\Big((1-p)\alpha + p\beta\Big) - [(1-p)h(\alpha) + ph(\beta)]$. Multiplying by the number of bits n gives (2). Eq. (3) follows from $\mathsf{H}(X|E) = \mathsf{H}(X) - I(X;E)$.

B. Proof of Theorem 2

The bit error probabilities for the $X \to X''$ channel are

$$\begin{aligned} \Pr[X''=1|X=0] &= \alpha(1-\beta_\mathrm{z})+(1-\alpha)\alpha_\mathrm{z} \\ \Pr[X''=0|X=1] &= \beta(1-\alpha_\mathrm{z})+(1-\beta)\beta_\mathrm{z} \end{aligned} \tag{16}$$

We want to impose the constraint that these probabilities are equal and then mimimize the bit error rate under this constraint. We use Lagrange multipliers formalism. We introduce the notation $\alpha_\mathrm{z}=x^2$ and $\beta_\mathrm{z}=y^2$, thus enforcing $\alpha_\mathrm{z},\beta_\mathrm{z}\geq 0$. The Lagrangian for this minimization problem is

$$\begin{aligned} L(x,y,\lambda) = {} & \alpha(1-y^2)+(1-\alpha)x^2+\lambda\big[\alpha(1-y^2)+(1-\alpha)x^2 \\ & -\beta(1-x^2)-(1-\beta)y^2\big] \end{aligned} \tag{17}$$

where λ is the Lagrange constraint multiplier. Note that the quantity to be minimized is the first expression in (16); we could equally well have taken the second expression, or some combination. Setting the derivatives of the Lagrangian to zero gives

$$\begin{aligned} \frac{\partial L}{\partial x} &= 2x(1-\alpha)+\lambda\big[2x(1-\alpha)+2x\beta\big]=0 \\ \frac{\partial L}{\partial y} &= -2y\alpha+\lambda\big[-2y\alpha-2y(1-\beta)\big]=0 \\ \frac{\partial L}{\partial \lambda} &= \alpha(1-y^2)+(1-\alpha)x^2-\beta(1-x^2)-(1-\beta)y^2=0. \end{aligned} \tag{18}$$

The first two lines of (18) simplify to

$$\begin{aligned} x=0 \text{ or } \lambda &= \frac{\alpha-1}{1-\alpha+\beta} \\ y=0 \text{ or } \lambda &= \frac{-\alpha}{1+\alpha-\beta} \end{aligned} \tag{19}$$

This leaves two possible solutions of the whole set of equations,

$$\begin{aligned} \alpha_\mathrm{z} &= \frac{\beta-\alpha}{1+\beta-\alpha}, \quad \beta_Z=0, \qquad \text{if } \beta\geq\alpha \\ \beta_\mathrm{z} &= \frac{\alpha-\beta}{1+\alpha-\beta}, \quad \alpha_Z=0, \qquad \text{if } \alpha\geq\beta \end{aligned}$$

Substituting α_z and β_z into (16) yields ε.

C. Proof of Lemma 5

In bit i we have the following conditional probabilities,

$$\begin{aligned} &\mathrm{Prob}[D_{1,i}\neq D_{2,i}|X_1=x_1,X_2=x_2] \\ &= \begin{cases} 2\alpha_\mathrm{d}(1-\alpha_\mathrm{d}) \text{ if } x_{1,i}=x_{2,i}=0 \\ 2\beta_\mathrm{d}(1-\beta_\mathrm{d}) \text{ if } x_{1,i}=x_{2,i}=1 \\ \alpha_\mathrm{d}(1-\beta_\mathrm{d})+(1-\alpha_\mathrm{d})\beta_\mathrm{d} \text{ if } x_{1,i}\neq x_{2,i} \end{cases} \end{aligned} \tag{20}$$

We compute $P_\mathrm{uneq} \overset{\mathrm{def}}{=} \mathrm{Prob}[D_{1,i}\neq D_{2,i}] = \mathbb{E}_{x_1x_2}\mathrm{Prob}[D_{1,i}\neq D_{2,i}|X_1=x_1,X_2=x_2] = \frac{1}{4}\sum_{x_{1,i}x_{2,i}}\mathrm{Prob}[D_{1,i}\neq D_{2,i}|X_1=x_1,X_2=x_2]$. Performing the summation and then simplifying the result yields (10). 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.

D. Proof of Theorem 7

$$\begin{aligned} I(\mathcal{E};D) &= \mathsf{H}(\mathcal{E})-\mathsf{H}(\mathcal{E}|D) && (21)\\ &= \mathsf{H}(D+\mathcal{N})-\mathsf{H}(\mathcal{N}) && (22)\\ &= \mathsf{H}(D+\mathcal{N}_\mathrm{av},D+\mathcal{N})-\mathsf{H}(\mathcal{N}_\mathrm{av},\mathcal{N}) && (23)\\ &= \mathsf{H}(D+\mathcal{N}_\mathrm{av})+\mathsf{H}(D+\mathcal{N}|D+\mathcal{N}_\mathrm{av}) \\ &\quad -[\mathsf{H}(\mathcal{N}_\mathrm{av})+\mathsf{H}(\mathcal{N}|\mathcal{N}_\mathrm{av})] && (24)\\ &= \mathsf{H}(D+\mathcal{N}_\mathrm{av})+\mathsf{H}(\mathcal{N}|\mathcal{N}_\mathrm{av}) \\ &\quad -[\mathsf{H}(\mathcal{N}_\mathrm{av})+\mathsf{H}(\mathcal{N}|\mathcal{N}_\mathrm{av})] && (25)\\ &= \mathsf{H}(D\oplus\mathcal{N}_\mathrm{av})-\mathsf{H}(\mathcal{N}_\mathrm{av}). && (26) \end{aligned}$$

E. Proof of Theorem 8

We have

$$\begin{aligned} \mathsf{H}(X|\mathcal{E}) &\geq \mathsf{H}(X|D\mathcal{N}) && (27)\\ &= \mathsf{H}(X|D)+\mathsf{H}(\mathcal{N}|XD)-\mathsf{H}(\mathcal{N}|D) && (28)\\ &= \mathsf{H}(X|D)+\mathsf{H}(\mathcal{N}|X_\mathrm{drifted})-\mathsf{H}(\mathcal{N}|D) && (29)\\ &\geq \mathsf{H}(X|D)+\mathsf{H}(\mathcal{N}|X_\mathrm{drifted})-\mathsf{H}(\mathcal{N}) && (30)\\ &= \mathsf{H}(X|D)-I(\mathcal{N};X_\mathrm{drifted}). && (31) \end{aligned}$$

In (27) we used that D and $\mathcal{N}$ together contain more information than $\mathcal{E}$. In (29) we used that $\mathcal{N}$ depends on X and D only through $X\oplus D$. Finally we take $\mathsf{H}(X)$ minus the whole inequality (27,31).

F. Proof of Theorem 9

$$\begin{aligned} \mathsf{H}(X|W\mathcal{E}) &= \mathsf{H}(X|W)+\mathsf{H}(\mathcal{E}|XW)-\mathsf{H}(\mathcal{E}|W) \\ &= \mathsf{H}(X|W)+\mathsf{H}(\mathcal{E}|X)-\mathsf{H}(\mathcal{E}|W) && (32)\\ &\geq \mathsf{H}(X|W)+\mathsf{H}(\mathcal{E}|X)-\mathsf{H}(\mathcal{E}) && (33)\\ &= \mathsf{H}(X|W)-I(\mathcal{E};X). && (34) \end{aligned}$$

In (32) we used the fact that W is a function of X. Finally we take $\mathsf{H}(X)$ minus the whole inequality derived above.

G. Proof of Theorem 10

We write

$$\mathsf{H}(X|\tilde{W}\hat{D}\mathcal{L})=\mathsf{H}(X\tilde{W}\hat{D}\mathcal{L})-\mathsf{H}(\tilde{W}\hat{D}\mathcal{L}). \tag{35}$$

Applying the chain rule again we expand these terms as

$$\begin{aligned} \mathsf{H}(X\tilde{W}\hat{D}\mathcal{L}) &= \mathsf{H}(X)+\mathsf{H}(\tilde{W}\hat{D}\mathcal{L}|X) \\ &= \mathsf{H}(X|W)+\mathsf{H}(W) && (36)\\ &\quad +\mathsf{H}(\hat{D}|X)+\underbrace{\mathsf{H}(\tilde{W}|\hat{D}X)}_{0}+\underbrace{\mathsf{H}(\mathcal{L}|\tilde{W}\hat{D}X)}_{\mathsf{H}(\mathcal{L}|\hat{X}_\mathrm{drifted})} \end{aligned}$$

and

$$\mathsf{H}(\tilde{W}\hat{D}\mathcal{L}) = \mathsf{H}(\hat{D})+\mathsf{H}(\tilde{W}|\hat{D})+\mathsf{H}(\mathcal{L}|\hat{D}\tilde{W}). \tag{37}$$

In (36) 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 (36) and (37) into (35).

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. 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.

[6] S. Golomb, "The limiting behavior of the Z-channel (Corresp.)," *IEEE Transactions on Information Theory*, vol. 26, no. 3, pp. 372–372, 1980.

[7] 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.

[8] 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.

[9] C. Bennett, G. Brassard, C. Crépeau, and M. Skubiszewska, "Practical Quantum Oblivious Transfer," in *CRYPTO*, 1991, pp. 351–366.

[10] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Cryptographic Hardware and Embedded Systems–CHES 2012*. Springer, 2012, pp. 283–301.

[11] S. Moser, P.-N. Chen, and H.-Y. Lin, "Error Probability Analysis of Binary Asymmetric Channels," 2010, intermediate report of project Finite Blocklength Capacity (proj.-no.: NSC 97-2221-E-009-003-MY3), funded by National Science Council, Taiwan.

[12] A. Schaller, B. Škorić, and S. Katzenbeisser, "On the systematic drift of physically unclonable functions due to aging," in *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices*. ACM, 2015, pp. 15–20.