Eliminating leakage in reverse fuzzy extractors

Citation for published version (APA):

Document license:
TAVERNE

DOI:
10.1109/TIFS.2017.2774500

Document status and date:
Published: 01/04/2018

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 17. Sep. 2023
Eliminating Leakage in Reverse Fuzzy Extractors

André Schaller®, Taras Stanko, Boris Škorić®, and Stefan Katzenbeisser, Senior Member, IEEE

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 is reverse fuzzy extractors, shifting the computational load of error correction toward 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 linked to the identity and the security problem (if the noise is data-dependent)). In this paper, we quantify the issue of leakage due to asymmetry of noise, leveraging the binary asymmetric channel (BAC) model. We further propose to concatenate two BACs to form a symmetric channel, as a solution that is able to eliminate such noise. Finally, we propose a modified reverse fuzzy extractor that does not leak via the error patterns even in the case of systematic drift of the PUF responses.

Index Terms—Error correction codes, authentication.

I. INTRODUCTION

In the past decade Physically Unclonable 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, which is sometimes referred to as ‘Physically Obfuscated Key’. In particular, 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, such as low-cost anonymous access tokens.

II. THE USE OF REVERSE FUZZY EXTRACTORS

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 device containing the PUF is assumed to be resource-constrained. In the key reconstruction phase the device needs to perform an ECC decoding step, which may be infeasible given the constraints. An elegant solution was proposed in [4], where it was shown how the ECC decoding can be securely outsourced to a more powerful second party. The scheme was dubbed ‘Reverse Fuzzy Extractor’. The most difficult HDS task for the device 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 low ‘drift’ in the values of their response bits over time, which is due to device ageing. This drift is characteristic to individual PUF
instances. A passive network 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, especially those that rely on the Reverse Fuzzy Extractor [7], [8], as the drift allows an 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.

A. New Contributions

This paper is an extension to our publication [9], where we presented an evaluation of the systematic drift of Physically Unclonable Functions due to aging and further analyzed leakage involved. This version extends our previous work with the following contributions:

- We adopt the Binary Asymmetric Channel (BAC) as a noise model and provide detailed numbers on the potential leakage caused by the asymmetry of the noise.
- We propose an approach to eliminate the leakage. We apply artificial asymmetric noise. This results in two concatenated BACs which together form a symmetric channel. Due to the symmetry the leakage is entirely eliminated.
- The introduction of artificial noise leads to a loss of channel capacity. We estimate this loss.
- Finally, we propose a modified Reverse Fuzzy Extractor Protocol, which is resistant against leakage even if involved PUF instances exhibit drift.

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 particular. 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 analyze 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

\[ h(p) = -p \log p - (1 - p) \log(1 - p). \]  

(1)

The Shannon entropy of a random variable \(X\) is denoted as \(H(X)\), 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: \(\alpha \equiv \Pr[X' = 1|X = 0]\) and \(\beta \equiv \Pr[X' = 0|X = 1]\). 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 physical 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 [10]. 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, noisy responses must be stabilized. This is done by employing a Fuzzy Extractor [11], which extracts the stable part of the PUF response and transforms it to a uniformly distributed value.

C. Fuzzy Extractors

Dodis et al. [11] introduced Fuzzy Extractors as a means to deal with the noise. Commonly, Fuzzy Extractors work in two phases, a generation phase \(\text{Gen}()\) performed upon enrollment and a reconstruction phase \(\text{Rec}()\) performed after each measurement. During \(\text{Gen}()\), a secret key \(K\) and a public Helper Data \(W\) are derived from a noisy PUF reference (enrollment) measurement \(X\). The algorithm \(\text{Rec}()\) 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 measurements of 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 (such as signal processing of the raw PUF data, or additional protection of the helper data, hashes of the key, quantities derived from the key, usage of the key). The Reverse Fuzzy Extractor is a two-party protocol which involves a prover, in possession of a (resource- constrained) PUF-enabled device, who wants to authenticate towards a computational powerful verifier. The description below is identical to the ‘Syndrome-Only’ Code Offset Method [11], [12] with the sole difference that syndrome decoding is outsourced to the verifier. A sequence diagram of the protocol is given in Figure 1.

\(^1\)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.
1) System Setup: The parties agree on a linear error correcting code \( C \) with message length \( m \) and codeword length \( n \). The encoding algorithm of \( C \) is \( \text{Enc}: \{0,1\}^m \rightarrow \{0,1\}^n \), and the algorithm for computing the syndrome is denoted as \( \text{Syn}: \{0,1\}^n \rightarrow \{0,1\}^{n-m} \). The code is chosen such that an efficient syndrome decoder \( \text{SynDec}: \{0,1\}^{n-m} \rightarrow \{0,1\}^n \) exists. The parties also agree on a key derivation function \( \text{KeyDeriv}: \{0,1\}^n \rightarrow \{0,1\}^m \).

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

3) Reconstruction: The prover performs a fresh measurement \( X' \in \{0,1\}^n \). He computes \( \Sigma = W \oplus \text{Syn}(X') \) and sends \( \Sigma \) to the verifier. The verifier computes the estimators \( \hat{X} = X' \oplus \Sigma \) and \( \hat{K} = \text{KeyDeriv}(\hat{X}) \).

Note that this protocol is extremely lightweight, as the prover only has to perform one \( \text{Syn} \) and one \( \text{KeyDeriv} \) operation. Note further that \( \Sigma = \text{Syn}(X \oplus X') \), due to the linearity of the code \( C \). Hence, if there is not too much noise, \( E \) is the error pattern that maps \( X' \) back to \( X \) and \( \hat{X} = X \) during reconstruction.

Finally, Table I provides an overview of the variables used throughout this paper and their respective meaning.

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 \( E \), during the reconstruction (Section II-D). For instance, imagine that for a single bit of the PUF response a 0 \( \rightarrow 1 \) transition is much more likely than a 1 \( \rightarrow 0 \) 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 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 \( \text{Pr}[X_i = 1] = p \). 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

\[
I(X; E) = n \left[ h \left( (1 - p) \alpha + p \beta \right) - (1 - p) h(\alpha) - p h(\beta) \right]
\]

and the entropy of \( X \) given \( E \) is

\[
H(X|E) = nh(p) - I(X; E).
\]

The proof is given in Appendix A. Note that setting \( \alpha = \beta \) in Lemma 1 gives \( I(X; E) = 0 \), i.e. if the noise is data-independent then an attacker learns nothing about \( X \) by observing \( E \).

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., \( T_i = \sum_{j=1}^{k} E^{(j)}_i \). The generalization of (2) then becomes

\[
I(X; E^{(1)} \ldots E^{(k)}) = I(X; T)
\]

\[
= H(T) - H(T|X)
\]

\[
= nH(T_i) - nH(T_i|X_i)
\]

where in the last line the index \( i \) is arbitrary. For the evaluation of \( H(T_i|X_i) \) and \( H(T_i) \) we need the corresponding probability distributions. For given \( X_i \), \( T_i \) is binomial-distributed, thus \( \text{Pr}[T_i = t|X_i = 0] = \binom{k}{t} \alpha^t (1 - \alpha)^{k-t} \) and \( \text{Pr}[T_i = t|X_i = 1] = \binom{k}{t} \beta^t (1 - \beta)^{k-t} \). This yields \( \text{Pr}[T_i = t] = (1 - p) \text{Pr}[T_i = t|X_i = 0] + p \text{Pr}[T_i = t|X_i = 1] \).
Table II lists values for $\mu$ and $\delta$, which were computed by considering pairs of enrollment and reconstruction PUF measurements and applying the BAC model accordingly. In particular, for a given PUF type, we randomly selected an enrollment measurement at 20°C and compared it with all reconstruction measurements taken at $-40\, ^\circ C$, 20°C and 80°C. Values for $\alpha$ and $\beta$ were computed by counting bit flips in the PUF measurement (see Section II-A). Once values for ($\alpha, \beta$) pairs were determined for each combination of PUF type and reconstruction temperature, corresponding $\mu$ and $\delta$ values were derived. For Ring Oscillator (RO) PUFs, D-flip-flop (DFF) PUFs (high temperature) and Latch PUFs (low temperature), large values of $|\delta|$ up to 0.08, 0.17 and 0.2 respectively can occur (at $\mu \approx 0.2$), while SRAM PUFs have very little asymmetry with $|\delta| = 0.07$, $\mu = 0.002$. These are the ‘raw’ values before reliable component selection has been applied, or other processing, e.g. repetition codes, that reduces $\mu$ and $\delta$. It is clear from Figure 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 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 measurement channel $X \rightarrow X'$. 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 $\Sigma = W \oplus \text{Syn}(X'')$ and sends $\Sigma$ to the verifier.
2. The verifier computes the error pattern $E = \text{SynDec}(\Sigma)$ and sends $E$ to the prover.
3. The prover computes the estimators $\hat{X} = X'' \oplus E$ and $\hat{K} = \text{KeyDeriv}(\hat{X})$.

We define the notation $\alpha_{X} = \Pr[X'_{X} = 1|X'_{X} = 0]$ and $\beta_{X} = \Pr[X'_{X} = 0|X'_{X} = 1]$ for the Z-channel bit flip probabilities.
correction. We now quantify how much 'worse' the channel

\[ \varepsilon \]

we have useful information about \( \varepsilon \) the mediated error pattern, but this comes at a cost: the additional \( \alpha \) 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.

Theorem 2: Let \( X \rightarrow X' \) be a given BAC with parameters \( \alpha, \beta \) (or \( \mu, \delta \)). Let \( X' \rightarrow X'' \) be a second BAC with parameters \( \alpha_z, \beta_z \) such that the combined channel \( X \rightarrow X'' \) is a BSC with bit error rate \( \varepsilon \). Then \( \varepsilon \) is minimized by the following parameter choice:

\[
\alpha \geq \beta (\delta \leq 0): \alpha_z = 0, \quad \beta_z = \frac{\alpha - \beta}{1 + \alpha - \beta} = \frac{2|\delta|}{1 + 2|\delta|} \\
\beta \geq \alpha (\delta \geq 0): \beta_z = 0, \quad \alpha_z = \frac{\beta - \alpha}{1 + \beta - \alpha} = \frac{2\delta}{1 + 2\delta}.
\]

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.

Of course the legitimate parties need to estimate the noise parameters \( \alpha, \beta \) of the BAC in order to be able to set \( \alpha_z, \beta_z \) 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.

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 \( X' \rightarrow X'' \) is than the original channel \( X \rightarrow X' \). First we show that the final noise parameter \( \varepsilon \) cannot be larger than the highest BAC parameter.

Corollary 3: The bit-error probability \( \varepsilon \) specified in Theorem 2 satisfies

\[
\varepsilon \in \left[ \frac{\alpha + \beta}{2}, \max(\alpha, \beta) \right] = [\mu, \mu + |\delta|]. \tag{5}
\]

Proof: Let \( \beta \geq \alpha (\delta \geq 0) \) w.l.o.g. From Theorem 2 we have \( \varepsilon = \frac{\mu - |\delta|}{1 - 2\mu} \). Obviously, \( \varepsilon \leq \mu + |\delta| \). Furthermore, \( \varepsilon = \frac{\mu + |\delta|}{1 + 2\delta} = \frac{\mu(1 + 2\delta) + |\delta| - 2\delta^2}{1 + 2\delta} = \mu + \frac{\delta(1 - 2\mu)}{1 + 2\delta} \geq \mu \). In the last step we used \( 0 \leq \mu \leq 0.5 \). Thus, we obtain \( \varepsilon \in [\mu, \mu + |\delta|] = [\alpha, \beta] \). 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 \( X \rightarrow X' \) has capacity \( C_{\text{BSC}} = 1 - h(\varepsilon) \), with \( \varepsilon = \frac{\mu + |\delta|}{1 + 2|\delta|} \) as specified in Theorem 2. The capacity of the BAC is given by (see e.g. [19]),

\[
C_{\text{BAC}} = \frac{\mu - |\delta|}{1 - 2\mu} h(\mu + |\delta|) - \frac{1 - \mu - |\delta|}{1 - 2\mu} h(\mu - |\delta|) \\
+ \log \left( 1 + 2 \frac{h(\mu + |\delta|) - h(\mu - |\delta|)}{1 - 2\mu} \right) \\
= 1 - h(\mu) + \frac{\delta^2}{\mu^2} + O(\frac{\delta^2}{\mu^2}). \tag{6}
\]

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

\[
C_{\text{BAC}} - C_{\text{BSC}} \approx |\delta|(1 - 2\mu) \log \frac{1 - \mu}{\mu} + O(\delta^2)
\]

\[
= |\delta| \log \frac{1}{\mu} + O(\delta \log \frac{1}{\mu}). \tag{7}
\]

Proof: Follows from Taylor-expanding the expressions for \( C_{\text{BAC}} \) and \( C_{\text{BSC}} \).

In Figure 4 we plot the capacity loss \( C_{\text{BAC}} - C_{\text{BSC}} \) relative to the original capacity \( C_{\text{BAC}} \). The 'raw' noise levels in PUFs (i.e., without reliable cell selection) for different PUF types. As shown in Table II, D-flip-flop PUFs can have high noise levels up to \( \mu = 0.2, \delta = 0.1 \) [5]. In this case and according to Figure 4, the Z-channel insertion would then lead to almost 40% capacity loss. In contrast, SRAM PUFs exhibit comparably little asymmetry with \( \mu = 0.07, \delta = 0.02 \), which results in less then 5% capacity loss. Depending on the context this may be acceptable. If not, the noise \( \mu, \delta \) can be reduced, as in Figure 4 we see that the capacity loss is less severe at low noise. Reduction of noise due to \( \mu, \delta \) asymmetry can be achieved by techniques such as reliable component selection and repetition codes. The optimal tuning of the parameters in the noise reduction techniques depends on the specific PUF properties.

The consequence of reduced channel capacity in practical scenarios lies in the fact that more PUF material is needed due
to this loss of channel capacity. In fact, the channel capacity is inversely proportional to the size of required PUF input bits.

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.

A. Drift Model

We adopt the bias-based PUF model proposed in [5]. We define the bias of a PUF cell \( i \) to be the probability that the cell ends up in a ‘1’ state and denote the bias 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: \( \hat{b}_i = \frac{x_i}{k} \in [0, \frac{1}{k}, \ldots, 1] \). A PUF is fully characterized by a vector of biases, \( \hat{b} = (\hat{b}_i)_{i=1}^n \). Similarly, \( \hat{b}'_i \) 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: \( \hat{b}'_i = \frac{x'_i}{l} \in [0, \frac{1}{l}, \ldots, 1] \).

The drift is modeled 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 \( b \) 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 [5], [20]), and that drift behavior is the same for all bits, we can express the transition probability for the entire PUF as

\[
\tau(b'|b) = \prod_{i=1}^{n} \tau_0(b'_i|b_i).
\]  

The function \( \tau_0 \) does not depend on the cell index \( i \). To estimate \( \tau_0 \) 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 \( \hat{b}_i \rightarrow \hat{b}'_i \) occurrences. Here the \( \hat{b}_i \rightarrow \hat{b}'_i \) 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 [18]. 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 operating the ASICs at an extreme temperature of \(+85^\circ\text{C}\) 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 \( t_0 \) for SRAM, latch, DFF and RO PUFs corresponding to different time intervals \( (t_0 \rightarrow t_1 \text{ and } t_0 \rightarrow t_2) \). (a) SRAM: \( t_0 \rightarrow t_1 \). (b) SRAM: \( t_0 \rightarrow t_2 \). (c) Latch: \( t_0 \rightarrow t_1 \). (d) Latch: \( t_0 \rightarrow t_2 \). (e) DFF: \( t_0 \rightarrow t_1 \). (f) DFF: \( t_0 \rightarrow t_2 \). (g) RO: \( t_0 \rightarrow t_1 \). (h) RO: \( t_0 \rightarrow t_2 \).

In Figures 5a, 5c and 5e we observe a diagonal ‘saddle’ between (0, 0) and (1, 1) for the \( t_0 \rightarrow t_1 \) data. This indicates that SRAM, latch and DFF PUFs have a stable bias over a short operating time.
The RO PUF (Figure 5g) 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 not unexpected, as ring oscillators can be used to generate random numbers as well.

For the transition $t_0 \rightarrow t_2$ we see a flattening of the ‘saddle’ for all PUF types (Figures 5b, 5d, 5f and 5h). This indicates, as expected, that there is a significant drift after a longer operation time. Note that not all 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 ($t = 1$). Hence, in practice FEs usually do not use fine-grained information about 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 PUF cells are taken into account. For this purpose, we regard cells as reliable, if they observe an enrollment bias $b_i \in [0, 0.05] \cup [0.95, 1]$. Although the intervals that define stable components are chosen somewhat arbitrarily, it turns out to be a workable choice. The model has only two parameters: $\alpha_d$, the probability of a $0 \rightarrow 1$ bit transition due to drift, and $\beta_d$, the probability of a $1 \rightarrow 0$ transition due to drift. The numerical values of these parameters slowly vary as a function of time. Table III lists the transition probabilities of reliable cells, based on the empirical data from the UNIQUE project, and Figure 6 shows the same data as bit error rates graphically.\footnote{Note that this figure incorporates fixed values, which have been found to be erroneous in [9].}

For SRAM and RO PUFs, the $t_0 \rightarrow t_2$ bit error rate is considerably higher than the $t_0 \rightarrow t_1$ 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.

1) Privacy: We first quantify the privacy leakage of the Reverse FE protocol caused by observation of the drift. A privacy leakage occurs due to device unique drift that is manifested in the public error patterns. Privacy-preserving protocols [7], [8] that leverage the Reverse FE therefore are vulnerable to a passive network attacker. By eavesdropping the communicated error patterns, the attacker is able to link multiple protocol executions to individual PUF instances.

Equation (11) gives the exact information he gains about the drift from observing a set of error patterns. Thus, the attacker is able to effectively undermine potential privacy-preserving mechanisms in place.

**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 $\alpha_d, \beta_d$ as defined above. Then the Hamming distance between $D_1$ and $D_2$ is binomial-distributed, with parameters $n$ and $P_{\text{uneq}}$, where

$$P_{\text{uneq}} = \frac{2\alpha_d + \beta_d}{2}(1 - \frac{\alpha_d + \beta_d}{2}).$$  \hspace{1cm} (10)

The proof is given in Appendix C.

The following corollary shows how the uniqueness of individual drifts can be quantized in terms of Hamming distance.

**Corollary 6:** 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 $\alpha_d, \beta_d$. Then the expected Hamming distance between $D_1$ and $D_2$ is $\mu_{\text{HD}} = nP_{\text{uneq}}$, and the variance is $\sigma_{\text{HD}}^2 = nP_{\text{uneq}}(1 - P_{\text{uneq}})$.

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

### Table III

<table>
<thead>
<tr>
<th>Period</th>
<th>Transition</th>
<th>SRAM</th>
<th>Cache</th>
<th>DFF</th>
<th>RO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0 \rightarrow t_1$</td>
<td>$0 \rightarrow 1$</td>
<td>0.0026 ± 0.0002</td>
<td>0.0025 ± 0.0007</td>
<td>0.0017 ± 0.0003</td>
<td>0.0076 ± 0.0055</td>
</tr>
<tr>
<td></td>
<td>$1 \rightarrow 0$</td>
<td>0.0029 ± 0.0001</td>
<td>0.0012 ± 0.0003</td>
<td>0.0019 ± 0.0004</td>
<td>0.0050 ± 0.0043</td>
</tr>
<tr>
<td>$t_0 \rightarrow t_2$</td>
<td>$0 \rightarrow 1$</td>
<td>0.0130 ± 0.0004</td>
<td>0.0222 ± 0.0135</td>
<td>0.0062 ± 0.0028</td>
<td>0.0350 ± 0.0199</td>
</tr>
<tr>
<td></td>
<td>$1 \rightarrow 0$</td>
<td>0.0091 ± 0.0004</td>
<td>0.0061 ± 0.0038</td>
<td>0.0029 ± 0.0010</td>
<td>0.0233 ± 0.0184</td>
</tr>
</tbody>
</table>

Fig. 6. Bit error rates of stable cells for various PUF types after $t_0 \rightarrow t_1$ and $t_0 \rightarrow t_2$ aging. The red line in each box indicates the median. The colored bottom and top of each box marks the 25th and 75th percentile. The height of a box displays the inter quartile range (IQR). The whisker’s extend 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.
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 by Schaller et al.

For the further analysis we introduce the following notation. Given multiple observations of the protocol run, we denote the set of observed error patterns as $E = (E_d)_{d=1}^k$, where $k$ is the number of observations. Similarly, we define $N = (N_a)_{a=1}^k$, with $E_d = D \oplus N_a$. We write $X_{\text{drifted}} = X \oplus D$.

**Theorem 7:** Let $N_{av} \in \{0, 1\}^n$ be the pattern obtained by averaging $N$: $N_{av} = \frac{1}{N} \sum N_a$. The amount of information about $D$ gained from observing $E$ is given by

$$I(E; D) = H(D \oplus N_{av}) - H(N_{av}). \quad (11)$$

**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 $E$ is the same as observing $D$ and $E$ (or, equivalently, $D$ and $N$). The $D$ can be used by the attacker as an identifier. In the case of data-dependent noise, $N_a$ leaks information about $X_{\text{drifted}}$. This too can be used by the attacker as an identifier.

2) **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 $E$ and the PUF key $K$ in general. Instead, we derive a bound on the mutual information between $E$ and $X$.

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

$$I(E; X) \leq I(D; X) + I(N; X_{\text{drifted}}). \quad (12)$$

**Proof:** see Appendix E.

The two leakage terms in Theorem 8 are very similar. The $I(N; X_{\text{drifted}})$ term is exactly the leakage shown in Figure 2, but now about $X_{\text{drifted}}$ instead of $X$, which is practically the same from a security point of view, since the attacker has access to $D$. 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 $a$, $\beta$ by $a_d$, $\beta_d$. The $I(D; X)$ is nonzero if the drift is asymmetric.

Note that, in contrast to the leakage $I(N; X_{\text{drifted}})$, 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 $E$.

**Theorem 9:** The leakage caused by observing $W$ and $E$ can be bounded as

$$I(WE; X) \leq I(W; X) + I(E; X). \quad (13)$$

**Proof:** See Appendix F.

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

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 modified protocol assumes a passive network adversary, who is able to observe the channel between the prover and the verifier and hence is able to capture the communicated error patterns $E$. In the modified Reverse Fuzzy extractor 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 memory in which it stores an estimated drift vector $\hat{D} \in \{0, 1\}^n$ and a list $L$ of up to $N_{\text{max}}$ 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 $\hat{X}_{\text{drifted}}$, and then shifts the result by the amount of $\hat{D}$. Taking the drifted value $\hat{X}_{\text{drifted}}$ 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 $\hat{W} = \text{Syn}(\hat{X}_{\text{drifted}})$.

A detailed description of our proposal is given below.

1) **System Setup:** The same as in Section II-D.

2) **Enrollment:** The same as in Section II-D. The enrolled helper data is $\hat{W} = \text{Syn}(X)$. In addition, the prover’s list $L$ is initialized to the empty string $\emptyset$, and $\hat{D}$ is initialized to the zero string.

3) **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 $\Sigma = \hat{W} \oplus \text{Syn}(Y')$ and sends $\Sigma$ to the verifier.

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

3) The prover computes $\tilde{X}_{\text{drifted}} = Y' \oplus \tilde{E}$ and the estimators $\hat{X} = \tilde{X}_{\text{drifted}} \oplus \hat{D}$ and $\hat{K} = \text{KeyDeriv}(\hat{X})$.
   a) If $\hat{K} = K$ then the prover performs the following actions.
      (a) Add the error pattern $\tilde{E} \oplus R$ to the list $L$. If necessary, the oldest entry in $L$ is discarded to make place.
on 20 reconstruction measurements, taken at time periods distance of zero indicates a perfect estimator $\hat{D}$. A Hamming distance of zero results in estimator $\hat{D}$ being accurate. The actual drift $D$ is based on 20 reconstruction measurements, taken at time periods $t_1$ & $t_2$. The percentage values depict the fraction of reconstruction measurements used for computing $\hat{D}$. Fractional Hamming distances are given as mean values over 40 PUF instances.

(b) If $L$ contains $N_{\text{max}}$ entries, check if there are bit positions that are ‘1’ in the majority 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 $\hat{W}$ by $\text{Syn}(\hat{X}_{\text{drifted}} \oplus e)$. Xor all entries in $L$ 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 $\hat{X}_{\text{drifted}} = X \oplus \hat{D}$. This gives

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

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

We checked the accuracy of the estimator $\hat{D}$ of $D$, by simulating the proposed protocol 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 $N_{\text{max}}/20$ for various PUF types at time periods $t_1$ and $t_2$. In particular, we evaluated the protocol on the following PUF types and their respective PUF response size: SRAM (8 kB), Latch (1 kB), DFF (1 kB) and RO (16 kB). The listed values are averages of 40 individual PUF instances. As expected, with increasing $N_{\text{max}}$, the accuracy of $\hat{D}$ improves up to the point where the entire data set is considered ($N_{\text{max}} = 20$), resulting in $\hat{D} \approx D$, i.e., a fractional Hamming distance close to zero. Note, that the accuracy of $\hat{D}$ is not exactly zero due to quantization noise. The results show that $\hat{D}$ deviates only by 2% from the actual drift in the worst case (considering only two measurements), for most of the PUFs. Only the RO PUFs, exhibiting very large asymmetry, of up to 10% deviation. If the prover implements sufficient memory to store even more error pattern instances, accuracy of estimator $\hat{D}$ can be further improved. In our experiments deviation of $\hat{D}$ is already at 0.5% when using 20 measurements for SRAM, Latch and DFF PUFs. For RO PUFs, more then 20 measurements must be stored, to limit deviation of $\hat{D}$ to under 5%.

Thus, a prover keeping track of only two error patterns already results in estimator $\hat{D}$ that is accurate enough, in order to mask the long-timescale drift, thus demonstrating the efficiency of the proposed protocol.

C. Security of the Proposed Protocol

As mentioned above, an adversary who observes $\tilde{E}$ learns nothing about $X_{\text{drifted}}$; 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 $\hat{D}, L$ together with $\hat{W}$ in memory. We study how the security is affected by this additional information.

**Theorem 10:** An adversary who observes the prover’s memory has the following amount of information about $X$,

$$I(X; \hat{W}D\hat{L}) = I(X; W) + [H(\hat{W}|\hat{D}) - H(W)]$$

$$+ I(\hat{D}; X) + [H(L|\hat{D}\hat{W}) - H(L|X_{\text{drifted}})].$$

In Theorem 10, the term $I(X; W)$ is the ‘known’ 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 KeyDer16v.
- The term $H(\hat{W}|\hat{D}) - H(W)$ is small, since $W$ equals the original helper data $W$ compensated by the drift.
- The term $H(L|\hat{D}\hat{W}) - H(L|X_{\text{drifted}})$ is negligible, since the noise stored in $L$ 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 domain of the physical memory of the prover device. In particular, an attacker who is able to access the data stored on the prover device, i.e., list $L$ and estimator $\hat{D}$, is able to extract as much information about $X$, as a passive network adversary, eavesdropping on protocol runs of the original FE protocol scheme.

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, rendering insecure any protocol that uses the Reverse FE in its original form. Furthermore, (b) there is privacy-sensitive leakage if the long-timescale PUF drift depends on the PUF value $X$. 
Our study of experimental data confirms the existence of asymmetric (data-dependent) drift in several types of PUFs.

We introduced 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. This solves the security problem; and (ii) drift compensation by storing the estimated drift and recent error patterns in the prover device. This solves the privacy problem.

The first modification turns the noisy channel \( X \rightarrow X' \) into an even more noisy channel \( X \rightarrow X'' \), compensating its previous asymmetry. The second modification ‘moves’ the drift problem from eavesdropping to the physical attack scenario, involving an attacker with access to the prover device. In the new scheme, an eavesdropper will not be able to identify the PUF device. Physical access to the prover device’s nonvolatile memory in the new scheme will yield as much information as eavesdropping in the original scheme.

Note that the noise parameters \( (\alpha, \beta) \) need to be estimated accurately, otherwise the second Binary Asymmetric Channel will not fully compensate the asymmetry; this results in residual leakage, given by Equation (4) with modified parameters. In practice it is difficult to do an entirely accurate estimate. Hence one has to perform a suitable amount of privacy amplification in the KeyDeri \( \text{iv} \) operation (Section II-D) in order to eliminate Eve’s knowledge.

**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, \( H(E_i) = h((1-p)\alpha + p\beta) \). Next we have \( H(E_i|X_i) = \Pr[X_i = 0]h(\alpha) + \Pr[X_i = 1]h(\beta) \). We obtain \( I(X_i; E_i) = H(E_i) - H(X_i|E_i) = h((1-p)\alpha + p\beta) - [(1-p)h(\alpha) + ph(\beta)] \). Multiplying by the number of bits \( n \) gives (2). Eq. (3) follows from \( H(X|E) = H(X) - I(X; E) \).

**B. Proof of Theorem 2**

The bit error probabilities for the \( X \rightarrow X'' \) channel are
\[
\begin{align*}
\Pr[X'' = 1|X = 0] &= \alpha (1 - \beta_z) + (1 - \alpha) \alpha_z \\
\Pr[X'' = 0|X = 1] &= \beta (1 - \alpha_z) + (1 - \beta) \beta_z
\end{align*}
\]
We want to impose the constraint that these probabilities are equal and then minimize the bit error rate under this constraint. We use Lagrange multipliers formalism. We introduce the notation \( \alpha_z = x^2 \) and \( \beta_z = y^2 \), thus enforcing \( \alpha_z, \beta_z \geq 0 \).

The Lagrangian for this minimization problem is
\[
L(x,y,\lambda) = \alpha (1 - y^2) + (1 - \alpha)x^2 + \lambda [\alpha (1 - y^2) + (1 - \alpha)x^2 - \beta (1 - x^2) - (1 - \beta)y^2]
\]
where \( \lambda \) is the Lagrange constraint multiplier. Note that the quantity to be minimized is the first expression in (15); we could equally well have taken the second expression, or some combination. Setting the derivatives of the Lagrangian to zero gives
\[
\begin{align*}
\frac{\partial L}{\partial x} &= 2x(1 - \alpha) + \lambda [2x(1 - \alpha) + 2x\beta] = 0 \\
\frac{\partial L}{\partial y} &= -2y\alpha + \lambda [-2y\alpha - 2y(1 - \beta)] = 0 \\
\frac{\partial L}{\partial \lambda} &= \alpha (1 - y^2) + (1 - \alpha)x^2 - \beta (1 - x^2) - (1 - \beta)y^2 = 0.
\end{align*}
\]
The first two lines of (17) simplify to
\[
\begin{align*}
x & = 0 \quad \text{or} \quad \lambda = \frac{\alpha - 1}{1 - \alpha + \beta} \\
y & = 0 \quad \text{or} \quad \lambda = \frac{-\alpha}{1 + \alpha - \beta}
\end{align*}
\]
This leaves two possible solutions of the whole set of equations,
\[
\begin{align*}
\alpha_z &= \frac{\beta - \alpha}{1 + \beta - \alpha}, \quad \beta_z = 0, \quad \text{if} \quad \beta \geq \alpha \\
\beta_z &= \frac{\alpha - \beta}{1 + \alpha - \beta}, \quad \alpha_z = 0, \quad \text{if} \quad \alpha \geq \beta
\end{align*}
\]
Substituting \( \alpha_z \) and \( \beta_z \) into (15) yields \( \varepsilon \).

**C. Proof of Lemma 5**

In bit \( i \) we have the following conditional probabilities,
\[
\Pr[D_{1,i} \neq D_{2,i}|X_1 = x_1, X_2 = x_2] = \begin{cases} 
2\alpha_d(1 - \alpha_d) & \text{if } x_{1,i} = x_{2,i} = 0 \\
2\beta_d(1 - \beta_d) & \text{if } x_{1,i} = x_{2,i} = 1 \\
\alpha_d(1 - \beta_d) + (1 - \alpha_d)\beta_d & \text{if } x_{1,i} \neq x_{2,i}
\end{cases}
\]
We compute \( P_{\text{uneq}} \) \( \equiv \Pr[D_{1,i} \neq D_{2,i}] = \sum_{x_1,x_2} \Pr[D_{1,i} \neq D_{2,i}|X_1 = x_1, X_2 = x_2] \sum_{x_{1,i},x_{2,i}} \Pr[D_{1,i} \neq D_{2,i}|X_1 = x_1, X_2 = x_2] \). 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 \( P_{\text{uneq}} \).

**D. Proof of Theorem 7**

\[
I(\mathcal{E}; D) = H(\mathcal{E}) - H(\mathcal{E}|D) = H(D + \mathcal{N}) - H(\mathcal{N})
\]
E. Proof of Theorem 8

We have
\[ H(X|E) \geq H(X|DN) \]
\[ = H(X|D) + H(N|XD) - H(N|D) \]
\[ = H(X|D) + H(N|X_{\text{drifted}}) - H(N|D) \]
\[ \geq H(X|D) + H(N|X_{\text{drifted}}) - H(N) \]
\[ = H(X|D) - I(N; X_{\text{drifted}}). \] (30)

In (26) we used that \( D \) and \( N \) together contain more information than \( E \). In (28) we used that \( N \) depends on \( X \) and \( D \) only through \( X \oplus D \). Finally we take \( H(X) \) minus the whole inequality (26,30).

F. Proof of Theorem 9

We write
\[ H(X|WE) = H(X|W) + H(E|XW) - H(E|W) \]
\[ = H(X|W) + H(E|X) - H(E|W) \]
\[ \geq H(X|W) + H(E|X) - H(E) \]
\[ = H(X|W) - I(E; X). \] (33)

In (31) we used the fact that \( W \) is a function of \( X \). Finally we take \( H(X) \) minus the whole inequality derived above.

G. Proof of Theorem 10

We write
\[ H(X|WD\hat{L}) = H(X|W) + H(D|X\hat{L}) - H(D). \] (34)

Applying the chain rule again we expand these terms as
\[ H(X|WD\hat{L}) = H(X) + H(W|D\hat{X}) \]
\[ = H(X|W) + H(W) \]
\[ + H(D|X) + H(W|D\hat{X}) + H(L|X_{\text{drifted}}) \]
\[ \leq H(L|X_{\text{drifted}}). \] (35)

and
\[ H(W|D\hat{L}) = H(D) + H(W|D\hat{I}) + H(L|D\hat{W}). \] (36)

In (35) we have used the fact that \( L \) is noise on \( X_{\text{drifted}} \) and therefore can depend at most on \( X_{\text{drifted}} \) itself. We substitute (35) and (36) into (34).

REFERENCES


André Schaller received the M.Sc. degree in computer science from the Technical University of Darmstadt in 2012, where he is currently pursuing the Ph.D. degree with the Security Engineering Group. His research interests include hardware-based and embedded security, with special focus on physically unclonable functions.

Taras Stanko received the M.Sc. degree in theoretical physics from Ivan Franko National University, Lviv, Ukraine, in 2011, and the Ph.D. degree in mathematics for industry in 2015. He is currently pursuing the Ph.D. degree with the Eindhoven University of Technology.

Boris Škorić received the Ph.D. degree in theoretical physics from the University of Amsterdam, The Netherlands, in 1999. From 1999 to 2008, he was a Research Scientist with Philips Research, The Netherlands, working first on display physics and later on security topics. In 2008, he joined the Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, as an Assistant Professor.

Stefan Katzenbeisser (S’98–A’01–M’07–SM’12) received the Ph.D. degree from the Vienna University of Technology, Austria. Since 2008, he has been a Professor with the Technical University of Darmstadt, and the Head of the Security Engineering Group. He has authored over 100 scientific publications. His current research interests include digital rights management, data privacy, software security, and cryptographic protocol design. He has served on the program committees of several workshops and conferences devoted to information security.