Variable-Length Bit Mapping and Error-Correcting Codes for Higher-Order Alphabet PUFs

Vincent Immler[1], Matthias Hiller[1], Qinzhi Liu[1,2], Andreas Lenz[3], and Antonia Wachter-Zeh[3]

[1] Fraunhofer Institute for Applied and Integrated Security (AISEC)
{vincent.immler,matthias.hiller,qinzhi.liu}@aisec.fraunhofer.de
[2] RWTH Aachen University
[3] Institute for Communications Engineering, Technical University of Munich (TUM)
andreas.lenz@mytum.de,antonia.wachter-zeh@tum.de

Abstract. Device-specific physical characteristics provide the foundation for Physical Unclonable Functions (PUFs), a hardware primitive for secure storage of cryptographic keys. So far, they have been implemented by either directly evaluating a binary output or by mapping outputs from a higher-order alphabet to a fixed-length bit sequence. However, the latter causes a significant bias in the derived key when combined with an equidistant quantization.

To overcome this limitation, we propose a variable-length bit mapping that reflects the properties of a Gray code in a different metric, namely the *Levenshtein* metric instead of the classical Hamming metric. Subsequent error-correction is therefore based on a custom insertion/deletion correcting code. This new approach effectively counteracts the bias in the derived key already at the input side.

We present the concept for our scheme and demonstrate its feasibility based on an empirical PUF distribution. As a result, we increase the effective output bit length of the secret by over 40% compared to state-of-the-art approaches while at the same time obtaining additional advantages, e.g., an improved tamper-sensitivity. This opens up a new direction of Error-Correcting Codes (ECCs) for PUFs that output responses with symbols of higher-order output alphabets.

Keywords: Physical Unclonable Functions, Fuzzy Extractor, Secrecy Leakage, Coding Theory, Quantization, Varshamov-Tenengolts (VT) Code.

1 Introduction

For a variety of applications, PUFs provide cryptographic keys with an increased level of security when compared to previous approaches, e.g., keys stored in non-volatile memory that can be extracted while the device is powered off. Most PUFs are implemented in silicon, such as the Ring Oscillator (RO) [1] or SRAM

PUF [2]. Other more specialized approaches include the Coating PUF [3] which additionally provides tamper-evidence, i.e., a property that is required to determine if the device has been physically tampered with. At their core, all PUFs output quasi-continuous physical measurement data that is processed by a quantization and error correction to generate reliable keys.

For the ease of implementation, most PUFs map these quasi-continuous values to a single-bit response, as it is the case for the RO PUF. However, this discards large portions of the information provided by the PUF response. It was shown, e.g., for the Coating PUF, that a multi-bit quantization step increases the output entropy and also facilitates a first error reduction step [3].

So far, the non-uniformly distributed input data of the quantization is mapped to symbols by a fixed-length bit mapping. Depending on the selected type of quantization this has several drawbacks. For equiprobable quantization, helper data vectors leak significant amounts of secret information and also the tamper-sensitivity is poor, i.e., physical changes in the underlying PUF structure may not necessarily cause a change in the output of the quantization [4]. For an equidistant quantization, the resulting binary sequence is heavily biased and causes secrecy leakage in the helper data of a subsequent ECC.

To address this issue, we follow the information-theoretical intuition of quantizing values with different probabilities of occurrence to binary sequences of varying length, i.e., values that occur more often are assigned a shorter binary representation and vice-versa. A good compression algorithm maps a non-uniform sequence to a shorter uniformly distributed sequence. Therefore, the output binary data is nearly unbiased and the underlying equidistant quantization does not leak secret information. Moreover, for tamper-evident PUFs, an equidistant quantization is more sensitive towards physical attacks [4].

Unfortunately, following this idea comes at the expense that a large body of previous work on error correction [5,6,7,8,9,10,11,12,13] can no longer be applied to the quantized bit sequence of a PUF. This is owed to the fact that if noise exceeds the tolerance of the quantization scheme, the *length* of the considered sequence changes. A change in length is either called an *insertion* if it gets longer, or a *deletion* if it gets shorter. For more advanced cases not specifically considered in the paper, they may occur also at the same time.

In contrast, commonly known ECCs are directed towards correcting *substitution* errors, typically by taking into account the Hamming distance of sequences. Since one insertion or deletion does not only affect the erroneous symbol itself, but also shifts all subsequent symbols, codes in the Hamming metric are not able to efficiently correct insertion or deletion errors.

The challenge therefore is to use codes capable of correcting errors that stem from variable-length bit mappings within the context of PUFs, i.e., they must address common design issues of PUF key derivation schemes such as reliability and secrecy leakage in the helper data. To do so, we leverage the properties of Varshamov-Tenengolts (VT) codes [14,15,16] that are able to correct insertion and deletion errors. In fact, we use a variation of the original VT codes that also covers substitution errors.

To further elaborate on our scheme, let us briefly introduce the PUF system model, as illustrated in Figure 1. The upper part represents the enrollment of the PUF, i.e., the point in time when it is initialized in a secure environment and helper data is created to enable later error correction. The lower part depicts the reconstruction in the field where the PUF key is extracted again to serve as secret input for cryptographic applications. As part of this processing chain, the PUF values are affected by noise which makes it necessary to compensate for this influence by suitable schemes, e.g., a combination of quantization and ECC. In this work, we focus on the specifics of this algorithmic part.

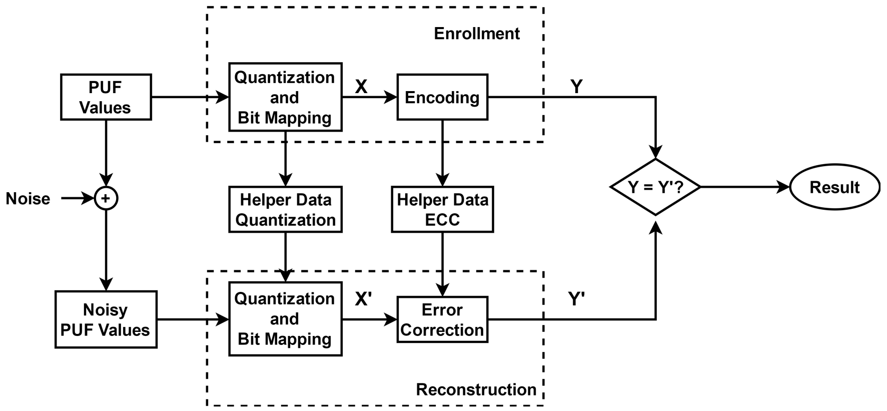


Fig. 1: PUF system model with enrollment and reconstruction. X is the quantized PUF response and Y the secret bit sequence. Added noise is denoted as $(\cdot)'$.

1.1 Contributions

In short, this work presents the following three contributions:

- A variable-length bit-mapping scheme that is well-adjusted in terms of the Levenshtein distance to the properties of an equidistant quantization.
- First application of codes with insertion and deletion error-correcting capability in the domain of PUFs including necessary code modifications.
- Practical design and comparison to state-of-the-art approaches, showcasing a gain of over 40% in effective output secret bits while at the same time improving tamper-sensitivity and ensuring sufficient reliability.

1.2 Organization

A brief outline of our paper is as follows. Related work is discussed in Section 2, while the required background information on insertion/deletion-correcting codes is reviewed in Section 3. Subsequently, we introduce our custom VT based key derivation scheme in Section 4. This new scheme is then evaluated in Section 5. Eventually, we conclude our work in Section 6.

1.3 Notation

Unless specifically noted otherwise, random variables and their distributions are represented by capital letters, whereas numbers and specific realizations of random variables are denoted as small letters. Subscripts refer to indices of vectors, and superscripts show the length of vectors (in either symbols or bit). $\mathcal{C}_{\mathrm{VT}}$ is the ECC and c stands for the n-bit codeword with m information bits and r parity bits.

For the helper data W^*, a quantized PUF response X^v with either superscript v as the symbol-wise length with alphabet size q or superscript n as length in bit, the mutual information between PUF response and helper data $\mathrm{I}(X^v; W^*)$ measures the information leakage. The min-entropy definition for $\tilde{\mathrm{H}}_\infty(X^v|W^*)$ is given in [6]:

$$\mathrm{I}(X^v; W^*) = \mathrm{H}(X^v) - \mathrm{H}(X^v|W^*) \leq v \cdot \log_2(q) - \tilde{\mathrm{H}}_\infty(X^v|W^*), \tag{1}$$

$$\tilde{\mathrm{H}}_\infty(X^v|W^*) = -\log_2\left(\mathop{\mathrm{E}}_{w^*}\left[\max_{x^v} \Pr_{X^v|W^*}[x^v|w^*]\right]\right). \tag{2}$$

2 State of the Art

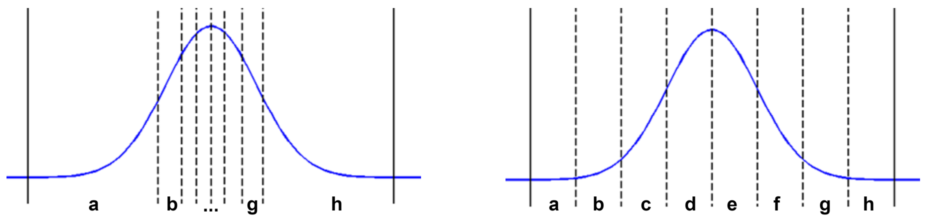
We align our work with two other domains. In Section 2.1, we discuss previous work on quantization schemes and bit mappings. Subsequently, in Section 2.2 we briefly consider other ECC proposals for PUFs and explain why they cannot be applied to our setting.

2.1 Quantization Schemes and Bit Mappings

A common approach for generating secret keys from PUFs with continuous output values is to apply an equiprobable quantization as in [3] or [17]. The Probability Density Function (PDF) over all analog PUF responses is divided into intervals of equal probability and each interval is mapped to a symbol from a higher-order alphabet as illustrated in Figure 2a. In order to decrease the probability of an erroneous quantization value, an offset is stored during enrollment that shifts the PUF response to the center of its corresponding quantization interval. However, as shown in [4], equiprobable quantization with these correcting vectors causes significant helper data leakage and requires precise knowledge of the distribution of the sampled PUF values. Hence, investigating other schemes is necessary.

Other equiprobable quantization schemes implement a partitioning scheme to avoid helper data leakage but again require precise knowledge of the distribution [18]. Also, for equiprobable approaches, tamper-sensitivity varies significantly due to the varying size of the quantization intervals [4]. Equidistant quantization intervals mitigate these effects but come at the downside of biased quantized PUF outputs. Here, the PDF is divided into intervals of equal width but different probability as shown in Figure 2b. As a consequence, a suboptimal assignment of the interval boundaries relative to the PDF only has a minor impact on the resulting entropy of the quantized output.

For both cases, the resulting symbols can be represented with a Gray code bit mapping, i.e., neighboring intervals differ only in a single bit position in terms of the Hamming distance. This results in a practical scheme for an equiprobable quantization, neglecting the challenge of precisely knowing the PDF. However, when combining equidistant quantization with fixed-length binary outputs and a linear fuzzy extractor scheme, significant amounts of secret information are leaked by the helper data due to the induced bias [19].



(a) Equiprobable quantization.

(b) Equidistant quantization.

Fig. 2: Visualization of equiprobable and equidistant quantization schemes.

2.2 Error-Correcting Codes for PUFs

A significant amount of work was carried out in the domain of PUFs ranging from formalizing PUFs [20] to generic ECC constructions, and protocols [21] in addition to analyses in terms of implementation and information efficiency [22,19].

As outlined before, previous work is mostly specifically tailored towards PUFs with a binary alphabet. The strong focus on these PUFs has been a valid requirement due to their large availability. While generally being suitable to provide a sufficient reliability, these schemes suffer from other shortcomings mostly related to helper data leakage that is caused by biased PUF data, as explained in [23]. Schemes targeting PUFs that provide higher-order alphabets must take these possible effects into account, too.

While lacking the opportunity to use existing ECC constructions, we still need to check if suitable ideas from the binary or fixed-length domain could be applied to our scenario, e.g., to prevent helper data leakage and bias. To remove this leakage, various debiasing schemes were proposed.

Index-Based Syndrome coding (IBS) [8] is a debiasing technique that also improves the reliability by indexing only reliable PUF response bits. However, the quantized input values all have the same reliability for equidistant quantization such that IBS is not applicable for the discussed scenario.

The scheme presented in [24] improves the von Neumann (VN) corrector [25]. For i.i.d. PUF response bits, pairs of consecutive zeros or ones occur with different probabilities, while pairs (1,0) and (0,1) have the same probability. However, the approach is intended for PUFs with small output alphabets. It evaluates groups of elements that occur with the same probability but differ in their sequence, such that an increasing number of elements decreases the probability of these equiprobable events. In [26], it was recently extended to ternary outputs using reliability information. However, it cannot be efficiently applied to higher-order alphabets. The multi-bit symbol approach in [27] is especially suited for PUFs with high bit error probabilities $> 20\%$. It is not explicitly designed for bias reduction but can also handle biased inputs efficiently as well. However, please note that it still has binary inputs and cannot compensate for insertion/deletion errors so that it cannot by applied under our constraints.

As a result, none of the discussed techniques provide a promising foundation to efficiently derive keys from PUFs with higher-order alphabets. To the best of our knowledge, the case of a variable-length bit mapping for PUFs has not been considered beforehand. We are aware of the threat of helper data manipulation attacks [28]. However, for the presented work, we are interested in discussing more fundamental properties of variable-length bit mappings and the corresponding ECCs.

3 Preliminaries

This section briefly introduces the two concepts that form the theoretical foundation of our proposed scheme. First, the Levenshtein distance is presented and its applicability to quantify the distortion caused by insertion/deletion errors is discussed. Second, VT codes are covered as a code class to deal with errors of this type.

3.1 Insertion/Deletion Errors and Levenshtein Distance

Let us briefly consider the following example: let $X = [1, 0, 1, 0, 1, 0, 1]$ be the designated bit sequence and $X' = [1, 1, 0, 1, 0, 1]$ a shorter received sequence where a deletion occurred at the second position of X. Since the Hamming distance is not defined between vectors of unequal length, one could artificially pad X' with a zero which results in $\mathrm{d_H}(X, [X', 0]) = 6$. This large distance highlights that it is impractical to rate deletions (and similarly, insertions) with the help of the Hamming metric.

To better reflect the nature of the error, Levenshtein [29] defined the distance $\mathrm{d_L}(X, X')$ as the smallest number of insertions, deletions, and substitutions that are required to transform X' into X. Hence, $\mathrm{d_L}(X, X') = 1$ for the given example. In the following, we review VT codes that form a class of codes that can correct errors in the Levenshtein metric.

3.2 VT Codes for Insertion/Deletion Error Correction

Varshamov-Tenengolts (VT) codes have been introduced to address insertion and deletion errors and correct a single insertion or deletion [15,30]. For a fixed integer $a \in \{0, \ldots, n\}$, a binary VT code of length n is defined as the set of all vectors $C^n = (c_1, c_2, \ldots, c_n) \in \{0,1\}^n$ such that:

$$\sum_{i=1}^{n} i \cdot c_i \equiv a \pmod{n+1}. \tag{3}$$

The integer a is called the checksum (or syndrome). VT codes are conjectured to be optimal in the sense that they have the largest cardinality of all single-deletion correcting codes [30]. The largest code sizes are obtained for $a = 0$. The size of the code for $a = 0$ is at least $\frac{2^n}{n+1}$ and its redundancy therefore at most $\lceil \log_2(n+1) \rceil$ bits. Please note that this basic construction is unable to correct substitutions and only works when the type of error is already known, i.e., the length of the received word must be provided.

The procedure to construct *systematic* VT-like codes according to [31] is as follows: For a binary input sequence $(x_1, \ldots, x_m)$, the corresponding codeword has the form $(c_1, \ldots, c_n)$ where $x_1 = c_{i_1}$, $x_2 = c_{i_2}, \ldots, x_m = c_{i_m}$, $1 \le i_1 < i_2 < \cdots < i_m \le n$. The bits c_k, where $k \notin \{i_1, i_2, \cdots, i_m\}$ are called parity bits. For a codeword of length n, the number of parity-check bits is $r = \lceil \log_2(n-1) \rceil + 1$ and they are located at positions $k = 2^l$, where $0 \le l \le r-2$, and at position n.

For M such that $2n \le M \le \min(n+2^{r-1}, 2^r)$, the parity-check bits $(p_1, \ldots, p_r)$ are chosen according to

$$\sum_{l=1}^{r-1} p_l \cdot 2^{l-1} + p_r \cdot n + \sum_{j=1}^{m} i_j \cdot x_j \equiv 0 \pmod{M}. \tag{4}$$

Please note, that "systematic" in this setting does not imply that the *first* m bits contain the information, they are distributed to positions which are not a power of 2 or equal to n. Extending this systematic encoding with the capability to also correct one substitution error comes at the expense of storing one additional redundancy bit.

In our PUF use case, only parts of the codewords are transmitted since the parity bits are stored as public helper data. The helper data is assumed not to be corrupted, so we can retrieve it without errors, similarly to [32]. However, the message bits may contain errors at unknown positions as they are drawn from the noisy PUF.

The standard systematic VT code cannot be employed in PUFs, because when recovering the response from the PUF, the positions where to insert the parity-check bits cannot be determined. It is therefore necessary to fully separate parity-check bits from the message containing secret information. This is explained in Section 4.2.

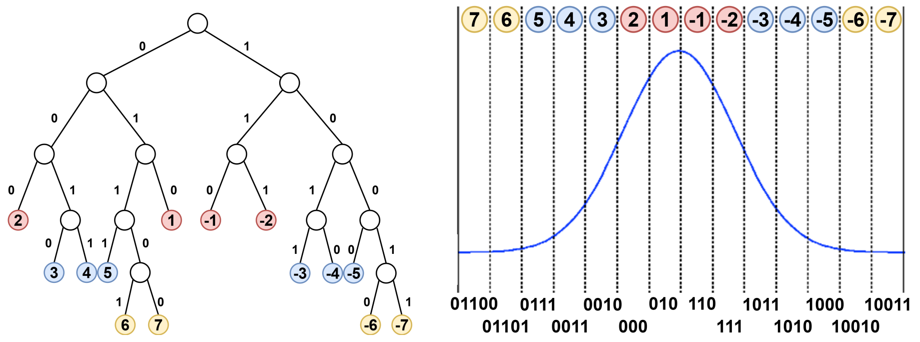
4 Variable-Length Bit Mapping and New VT-like Code

We first introduce the variable-length bit mapping of equidistant quantization intervals and discuss how to encode them into VT-like codewords.

4.1 Variable-Length Bit Mapping for Equidistant Quantization

Ideally, the bit mapping is such that the obtained sequence is not biased, i.e., the 1s and 0s are uniformly distributed. In addition, the bit mapping should support the subsequent error correction in terms of low distance changes from one to another quantization interval. At the same time this improves tamper-sensitivity, as errors that result in a larger distance to the designated value are almost certainly caused by a physical attack and therefore – as part of its intended purpose – should cause the device to fail.

To achieve low distance changes for neighboring intervals in Hamming distance, i.e., $\mathrm{d_H} = 1$, one would use a Gray code [33]. However, it cannot be applied in our case, since this scheme only works for fixed-length bit mappings. Another disadvantage of fixed-length bit mappings is that some patterns of 1s and 0s would occur more likely, i.e., cause a bias. To overcome these limitations, we propose a new variable-length bit mapping scheme, as shown in Figure 3b.



(a) Tree for variable-length bit mapping. (b) Resulting bit assignment.

Fig. 3: Proposed variable-length bit mapping for equidistant quantization.

In order to preserve the entropy of the quantization, i.e., when mapping its symbols to the binary domain, the quantization procedure requires a uniquely decodable code, e.g., it should be prefix-free. Therefore, we build a binary tree to explicitly assign symbols to a variable-length bit mapping that differs only in $\mathrm{d_L} = 1$ for neighboring intervals. Hence, it is the Levenshtein counterpart to the Gray code. Notice that a Huffman code is not an eligible candidate here as it neither ensures a debiasing characteristic due to the lack of same-probability of 0s and 1s, nor is the constraint of $\mathrm{d_L} = 1$ for neighboring intervals considered.

The example displayed for 14 intervals of Figure 3a is explained by following the conventions of graph theory. Let $\mathcal{G} = (\mathcal{V}; \mathcal{E})$ be the graph $\mathcal{G}$, whereas $\mathcal{V}$ represents the set of vertices and $\mathcal{E}$ the set of edges. The effective vertices are numbered from ± 1 to ± 7 to indicate the vertices's corresponding quantization interval to the left and right of the PDF's mean.

This construction follows the principle of a prefix-free code, where each effective vertex is connected to only one other vertex by one edge. For the resulting symbols of adjacent quantization intervals, the desired distance of $\mathrm{d_L} = 1$ is achieved. By traversing the graph either to the left or right, bit 1 or 0 is incorporated in the pattern. Unfortunately, we have not yet found a way to generalize this construction. The resulting bit mapping for the case of 14 intervals as represented by Figure 3b is therefore given in Table 1.

Table 1: Example for variable-length encoding with Levenshtein distance 1 between adjacent intervals. The colors are matched to Figure 3b.

Symbol	7	6	5	4	3	2	1	-1	-2	-3	-4	-5	-6	-7
Binary	01100	01101	0111	0011	0010	000	010	110	111	1011	1010	1000	10010	10011

The new bit mapping is well-suited for the application based on the following perspective:

- As long as the input distribution is symmetric, 0s and 1s are balanced, since equally probable intervals have an equal number of 1s and 0s.
- It fulfills the requirement that adjacent intervals only differ by one insertion/deletion/substitution error, i.e., adjacent intervals have $\mathrm{d_L} = 1$.
- It is prefix-free, i.e., it preserves the information provided by the quantization but with less redundant bits compared to a fixed-length bit mapping.
- It has a debiasing property, i.e., more probable symbols are assigned shorter bit mappings and less probable symbols are assigned longer bit mappings.

4.2 Systematic VT-Like Code Construction for PUFs

This section introduces a code to address a single insertion, deletion or substitution error that originates to a quantization error and subsequently stems from the bit mapping as introduced in Section 4.1. We propose a VT-like code construction for the situation that the parity-check bits are not transmitted within the input bit stream and are thus error-free. Our code construction is as follows:

$$\mathcal{C}_{\mathrm{VT}} := \Big\{(x_1, \cdots, x_m, p_1, \cdots, p_r) : \sum_{i=1}^{m} i x_i + \sum_{j=1}^{r} 2^{j-1} p_j \equiv 0 \pmod{2m+1}\Big\}, \tag{5}$$

where m information bits and r parity-check bits together for a codeword of length $n = m + r$. The redundancy of this code construction is $\lceil \log(2m + 1) \rceil$ and smaller than the redundancy of the systematic construction from [31]. In the following, we will show how $\mathcal{C}_{\mathrm{VT}}$ can correct one deletion, insertion, or substitution error. The decoding procedure is similar to the decoding of classical VT codes [30].

First consider an example with a single deletion. Assume that the π-th bit in the original bit sequence was deleted, which has λ_0 0s to the left of it, ρ_0 0s to the right of it, λ_1 1s left of it and ρ_1 1s right of it. Therefore, $\pi = 1 + \lambda_0 + \lambda_1$. Let ω be the Hamming weight the received bit stream, i.e., $\omega = \lambda_1 + \rho_1$. Evaluating the sums in Equation 5, the deficiency $\varDelta$ of the new checksum compared to the original one is

$$\varDelta = -(\pi \cdot x_\pi + \sum_{i=\pi+1}^{m} x_i) \pmod{(2 \cdot m + 1)} \tag{6}$$

When a 1 was deleted, the checksum deficiency is given by

$$\begin{aligned}
\varDelta &= -(\pi + \rho_1) && (7)\\
&= -(1 + \lambda_0 + \lambda_1 + \rho_1) && (8)\\
&= -(1 + \lambda_0 + \omega) && (9)\\
&\equiv 2 \cdot m + 1 - (1 + \lambda_0 + \omega) \pmod{2 \cdot m + 1} && (10)
\end{aligned}$$

To recover the initial input, one needs to insert a 1 at the right side of λ_0 0s in the received sequence. When a 0 was deleted, the new checksum is ρ_1 less than the original, i.e., $\varDelta = 2 \cdot m + 1 - \rho_1$. To recover, one needs to insert a 0 on the left side of ρ_1 1s. The case for insertion errors can be solved in a similar manner.

For substitution errors, the error pattern where the 0 flips to 1 gives a deficiency $\varDelta$ of the position number, i.e., π. Vice-versa, if 1 changes to 0, the deficiency $\varDelta$ is the value of $2m + 1 - \pi$. The range of values for the checksum deficiency $\varDelta$ for insertion, deletion, and substitution errors is given in Table 2.

Table 2: Checksum Deficiency $\varDelta$ vs. Error Pattern

Error Type	Error Pattern	$\varDelta$	Range of $\varDelta$
Insertion	insert 0	ρ_1	$[0, \omega]$
Insertion	insert 1	$\pi + \rho_1 = \omega + \lambda_0$	$[\omega, m + 1]$
Deletion	delete 0	$-\rho_1 + 2m + 1$	$[2m + 1 - \omega, 2m] \cup \{0\}$
Deletion	delete 1	$-\rho_1 - \pi + 2m + 1$	$[m + 1, 2m - \omega]$
Substitution	flip 0 to 1	π	$[1, m]$
Substitution	flip 1 to 0	$2m + 1 - \pi$	$[m + 1, 2m]$

The table shows that the range of the two cases of insertions overlap in ω. The error correction here can be explained as follows: for an insertion error, if $\varDelta = \omega$,

Algorithm 1: VT-like Systematic Decoding Algorithm for PUFs

```
Data:
$l_{\mathrm{I}}$ = (Length information)
$\Delta$ = (Checksum deficiency)
$X'$ = (noisy PUF response)
$m'$ = (bit length for reference PUF response)
Result: $Y'$ = (corrected secret bit sequence)
if $m' \equiv l_{\mathrm{I}} \pmod 3$ then
    /* substitution error or error-free,i.e., $m' = m$ */
    if $\Delta = 0$ then
        No error ;                                   // $Y' \leftarrow X'$
    else
        if $\Delta > m'$ then
            $X'[2m' + 1 - \Delta] = 1$ ;             // substitution error from 1 to 0
        else
            $X[\Delta] = 0$ ;                        // substitution error from 0 to 1
        end
    end
    $Y' \leftarrow X'$
else if $m' + 1 \equiv l_{\mathrm{I}} \pmod 3$ then
    /* deletion error, i.e., $m' = m - 1$ */
    if $\Delta = 0$ then
        $Y' \leftarrow X'$ with 0 inserted at the end
    else
        if $\Delta > 2 \cdot m' + 3 - \omega$ then
            insert 0 at left side of $\rho_1$ 1's on the right ;   // $\rho_1 = 2m' + 3 - \Delta$
        else
            insert 1 at right side of $\lambda_0$ 0's on the left ; // $\lambda_0 = 2m' + 2 - \omega - \Delta$
        end
        $Y' \leftarrow X'$
    end
else
    /* insertion error, i.e., $m' = m + 1$ */
    if $\Delta = 0$ then
        $Y' \leftarrow X'$ with 0 deleted at the end
    else
        if $\Delta > \omega$ then
            delete 1 at the right side of $\lambda_0$ 0's on the left ;   // $\lambda_0 = \Delta - \omega$
        else
            delete 0 at the left side of $\rho_1$ 1's on the right ;      // $\rho_1 = \Delta$
        end
        $Y' \leftarrow X'$
    end
return $Y'$
```

there is either a 0 or 1 inserted in the beginning. For this case, we delete the first bit to correct the insertion error. Algorithm 1 shows the decoding procedure for our proposed VT-like code construction. It generalizes the systematic decoding process of the previously discussed example.

In Algorithm 1, l_{I} denotes the length information $m \pmod 3$ which is stored as helper data. It allows to identify the error type. Recall that X' is the output of the measured PUF values and Y' is the corrected secret.

We increase argument of the modulo operation to $2m+1$ to also guarantee substitution error correction. If we only have insertion or deletion errors, we use the following code definition which has one bit less redundancy:

$$\{(x_1 \cdots x_m, p_1 \cdots p_r) | \sum_{i=1}^{m} i \cdot x_i + \sum_{j=1}^{r} 2^{j-1} \cdot p_j \equiv 0 \pmod{m+1}\}. \tag{11}$$

4.3 Helper Data

Our coding scheme stores two types of helper data, the length indicator information l_{I} and the parity bits p^r. We could also directly store the length m in l_{I}. However, this significantly reduces the number of possible sequences which is equivalent to a large helper data leakage. The VT-like code can only correct a single insertion, deletion or substitution such that we only need to correctly indicate whether the length of the sequence was increased by one, decreased by one or remained the same. This can also be represented by $m \pmod 3$ which reveals less information than providing the precise length.

In addition, the parity information is stored as helper data according to [32]. It was shown in [34] that the other linear schemes have a higher efficiency for the Hamming metric while the parity approach in [32] is less efficient and has a higher secrecy leakage. However, the other schemes apply an XOR operation between parts of the helper data and the PUF response. An insertion or deletion error destroys the mapping such that error correction is no longer possible. This makes the parity approach currently the only applicable scheme.

4.4 Toy Size Example

In the following toy example, we demonstrate the encoding and decoding of our VT-like code. Based on PUF nodes with $x^8 = [5, 4, -3, -6, 7, -1, 2, 4]$. The symbols are encoded according to the bit mapping presented in Section 4.1, i.e.,

$$\mathrm{enc}(x^8) = [(0111), (0011), (1011), (10010), (01100), (110), (000), (0011)]. \tag{12}$$

Afterwards, 4 symbols are combined to one VT codeword. The first 4 symbols are encoded to a binary sequence of length 17. Therefore $l_{\mathrm{I}}(x^4) = 17 \equiv 2 \pmod 3$. The left half of Equation 5 is

$$\sum_{i=1}^{17} i\,x_i = 2+3+4+7+8+9+11+12+13+16 = 85 \equiv 15 \pmod{35}. \quad (13)$$

The parity bits are a binary representation of $35 - 15 = 20$, so $p^6 = (010100)$. For the second part of the PUF response, we analogously calculate the helper data $l_{\mathrm{I}} = 15 \equiv 0 \pmod 3$ and $p^6 = (001111)$.

To demonstrate deletion and insertion error correction, let us assume that during reconstruction one quantization error occurred in the third symbol and another one in the seventh symbol, such that $x'^8 = [5, 4, \mathbf{-2}, -6, 7, -1, \mathbf{3}, 4]$. Therefore the third symbol is encoded to (111) instead of (1**0**11), which corresponds to one deletion error. Computing $l_{\mathrm{I}}(x'^4) = 1 \equiv 16 \pmod 3$ shows that the one bit was deleted:

$$\Delta = \sum_{i=1}^{m} i\,x'_i + \sum_{j=1}^{r} 2^{j-1} p_j = 81 + 20 = 101 \equiv 31 \pmod{33+2}. \quad (14)$$

$\Delta = 2\cdot(16+1)+1-\rho_1$, therefore we have $\rho_1 = 4$ and insert 0 on the left of 4 1s in the right. Thus, we were able to detect the position of the deletion and correct the error. For the second half, let us assume that the third symbol shifted from 2 to 3 such that (00**1**0) is forwarded instead of (000). Now $l_{\mathrm{I}}(x'^4) = 1$. Since $\mathrm{I}(x^4) = 0$, one insertion occurred. $\Delta = 13$, so according to line 28 of Algorithm 1, we delete the 1 at the right side of $13 - 7 = 6$ 0s.

5 Evaluation

To allow a fair comparison to the state of the art, the results in this section have been simulated according to the scenario in [3]. We therefore used the following parameters: The device contains 128 PUF nodes with Gaussian distributed PUF responses with $\mu = 1.8\cdot 10^{-13}$ and $\sigma = 3.6\cdot 10^{-15}$. Individual measurements of the nodes are affected by Gaussian distributed, mean-free noise with $\sigma_{\mathrm{N}} = 2\cdot 10^{-16}$.

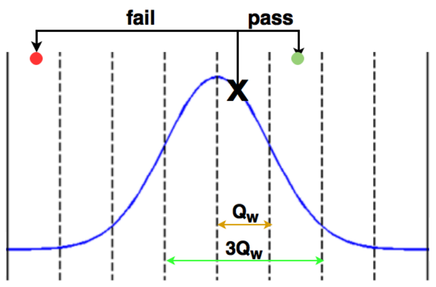
5.1 Reliability

In the following, two mechanisms are considered to improve the reliability of the PUF system. First, we evaluate the effects of the quantization. Afterwards, the specifics of the VT-like code are analyzed in terms of number of secret bits and reliability.

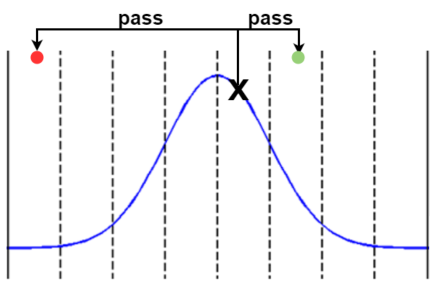
Error Reduction by Quantization As a baseline, we first evaluate the performance of a system that only relies on a quantization without any further error correction or leakage mitigation steps. Following [4], the equidistant quantization is applied to the PUF response of each individual node. The width of the quantization intervals is set to

$$Q_w = 2 \cdot y \cdot \sigma_{\mathrm{N}}. \tag{15}$$

As mentioned beforehand, by storing a helper data vector, the quantization scheme itself has an error tolerance of $[-y \cdot \sigma_{\mathrm{N}}, +y \cdot \sigma_{\mathrm{N}}]$, i.e., as long as the error does not exceed this interval no error will occur. Here, y is a parameter that determines the reliability. This is illustrated in Figure 4a with a yellow arrow indicating the interval Q_w. Later, we will combine this quantization with Reed–Solomon (RS) codes with different code parameters [35].



(a) Quantization and VT-like code as ECC.



(b) Quantization and RS code as ECC.

Fig. 4: Comparison of equidistant quantizations with differing subsequent ECCs. In 4a, only shifts by small errors are correct, as it is the preferred case to improve tamper-sensitivity. In contrast, 4b corrects any shift to arbitrary intervals, as long as the overall error-threshold of the RS code is not exceeded.

Error Reduction by VT-like Code Figure 4a illustrates the difference of the noise tolerance between the pure quantization and the combination of quantization with error correction. After error correction using the VT-like code, the noise tolerance has tripled to $3 \cdot Q_w$ for one value. Therefore, same values of y now offer a much better reliability compared to a pure quantization.

However, for each segment of nodes still only one error can be corrected due to the properties of the constructed code. This limitation is preferred, as a physical attack which causes a large increase in Levenshtein distance from the reference value should *not* be corrected. Heavily distorted measurement values occur from noise only with small probability, so multiple errors outside of the $[-y \cdot \sigma_{\mathrm{N}}, +y \cdot \sigma_{\mathrm{N}}]$ interval should cause the system to fail, thereby improving tamper-sensitivity.

In the following, we add a hat $(\hat{\cdot})$ to probabilities that refer to corrected values after the VT-decoding. We calculate the error probability P_n of a node by integrating over the PDF of the noise. Then we apply the VT-like code for error correction to obtain the corresponding error probability for a segment, if more than one node is corrupted with $\mathrm{d_L} = 1$. Finally, for an error-free device, all of its segments must be correct. The node error probability is calculated by the PDF of a Gaussian distribution with $\mathcal{N}(\mu, \sigma)$ as follows:

$$P_n = 1 - \int_{-y\cdot\sigma_\mathrm{N}}^{+y\cdot\sigma_\mathrm{N}} \mathcal{N}(0, \sigma_\mathrm{N}).$$

Without error correction, a segment with m nodes will pass the authentication process only if all its nodes are quantized correctly. This corresponds to a segment error probability P_s of

$$P_s = 1 - (1 - P_n)^m. \tag{16}$$

In this paper, the aim is to correct the error when the encoded value shifts into adjacent intervals. Hence, per segment, only one node with $\mathrm{d_L} = 1$ must be corrected. The probability $\hat{P}_n$ that a single node is correct after correction is:

$$\hat{P}_n = 1 - \int_{-3\cdot y\cdot\sigma_\mathrm{N}}^{+3\cdot y\cdot\sigma_\mathrm{N}} \mathcal{N}(0, \sigma_\mathrm{N}). \tag{17}$$

The error probability $\hat{P}_s$ after VT error correction is

$$\hat{P}_s \leq 1 - \Big(m(1 - P_n)^{m-1}(P_n - \hat{P}_n) + (1 - P_s)\Big) \tag{18}$$

$$= 1 - \Big(m(1 - P_n)^{m-1}(P_n - \hat{P}_n) + (1 - P_n)^m\Big) \tag{19}$$

The probability in (17) assumes that only adjacent intervals differ in one bit, i.e., a single insertion/deletion/substitution error. However, in the process of building the codebook, one cannot avoid that nearby intervals other than the adjacent ones also differ in only one bit.

Hence, the probability of the analytically computed error rate upper bounds the error probability and simulated results should slightly outperform the calculations. This difference can be practically observed, whereas the margin is larger for a higher error-rate and smaller for a lower error-rate. For a device with ν segments, the overall device error probability $\hat{P}_d$ is finally given by

$$\hat{P}_d = 1 - (1 - \hat{P}_s)^\nu. \tag{20}$$

If no error correction is carried out, $\hat{P}_d$ and $\hat{P}_s$ will be replaced by P_d and P_s.

As listed in Table 3, we observe for a device with 128 nodes that increasing y leads to an improved reliability at the expense of loss in entropy and shortened length of the bit sequence. Therefore, a designer's goal is to maximize the number of secret bits while meeting the reliability requirement.

Table 3: Effect of varying parameters of the quantization and resulting data for entropy, length of bit mapping, and reliability. The entropy is given in bits per node.

Number of Intervals	min Entropy	Shannon Entropy	Bits per Node	Bits per Device	97% Confidence Interval	P_d
12 ($y = 4.95$)	2.26	2.92	3.27	419	$[406, 430]$	9.5×10^{-5}
14 ($y = 4.24$)	2.47	3.13	3.36	430	$[417, 443]$	2.8×10^{-3}
16 ($y = 3.71$)	2.65	3.33	3.51	449	$[433, 466]$	2.6×10^{-2}
18 ($y = 3.30$)	2.81	3.49	3.73	478	$[457, 500]$	1.2×10^{-1}
20 ($y = 2.97$)	2.96	3.64	3.92	502	$[482, 517]$	3.1×10^{-1}

5.2 Information Leakage caused by ECC

To determine the amount of leakage between encoded sequence X^v helper data $W = (L_\mathrm{I}, P^*)$, we select one of our later results from Table 4 that meets the reliability requirements and has the largest number of effective secret bits. For other selected parameters, the calculation is similar.

The first source of leakage is caused by the stored length information l_I. It is stored for each segment and may have 3 possible values only. Therefore $\mathrm{I}(X^v; L_\mathrm{I})$ is considered as worst-case if rounded-up, i.e.,

$$\mathrm{I}(X^v; L_\mathrm{I}) \leq \mathrm{H}(\mathrm{L_I}) \leq \lceil \log_2(3) \rceil = 2\,\mathrm{bits}$$

The second source of leakage is based on the parity bits P^* of the VT code. For a segment with $v = 128$ node values, the maximum entropy of these parity bits is therefore considered as information leakage $\mathrm{I}(X^v; P^*)$. Please note, for the subsequent calculation, the maximum length of the segment is used as upper bound for the leaked bits. For the specific example, the code size determines the maximum entropy, i.e., here, resulting in the size of P^*. The remaining multiplicative factor of 2 and additive component + 1 is due to the structure of the code, cf. Equation (5):

$$\begin{aligned} \mathrm{I}(X^{128}; P^*) &\leq \mathrm{H}(P^*) && (21)\\ &\leq \lceil \log_2(2m + 1) \rceil && (22)\\ &= \lceil \log_2(2 \cdot 5 \cdot 128 + 1) \rceil && (23)\\ &= 11\,\mathrm{bits} && (24) \end{aligned}$$

Hence, the overall number of leaked bits based on a worst-case assumption is

$$\mathrm{I}(X^{128}; W^*) \leq 2 + 11 = 13\,\mathrm{bits} \qquad (25)$$

Concerning the min-entropy that is extracted on average from a device, we consider each node with $y = 4.94$ (resulting in 12 quantization intervals) which leads to a min-entropy of 2.26 bit per node, according to Table 3. This gives

$$\tilde{\mathrm{H}}_\infty(X^v) = 2.26 \cdot 128 = 289.3\,\text{bits} \tag{26}$$

Hence, for a device with 128 nodes, the number of overall effective secret bits is

$$\tilde{\mathrm{H}}_\infty(X^v) - \mathrm{I}(X^v; W^*) = 289.3 - 13 = 276.3\,\text{bits} \tag{27}$$

5.3 Comparison of Fuzzy Commitment and VT-like Codes

In the following, we compare a fuzzy commitment scheme based on an RS code and our VT-like code. The results for the RS code are given in Table 4. For the VT-like code, the results are summarized in Table 5. In either case, we make use of the min-entropy per node that we obtain from the quantization histogram as listed in Table 3. The values for y range from 3 to 5 and result in 2.26 to 2.96 bits of min-entropy per node.

Table 4: Evaluation of RS codes for PUFs with $v = 128$ output symbols. P_n and P_d are node and device error probabilities. Effective secret bit already account for the information leakage of the helper data.

y	z	RS Code Parameters	P_n (before RS)	P_d (before RS)	$\hat{P}_n$ (after RS)	$\hat{P}_d$ (after RS)	Effective Secret Bits
5	8	(15,13,3)	5.73×10^{-7}	7.34×10^{-5}	4.60×10^{-12}	4.79×10^{-10}	≈ 192
3.71	8	(15,11,5)	2.05×10^{-4}	2.59×10^{-2}	7.83×10^{-10}	6.89×10^{-8}	≈ 178
3	4	(31,23,8)	2.67×10^{-3}	2.90×10^{-1}	3.72×10^{-9}	3.42×10^{-7}	≈ 195

Table 5: Evaluation of error probability and information leakage for the proposed VT-like code. P_s and P_d are segment and device error probabilities for a PUF with $v = 128$ output symbols. Leakage $I(X^v; W^*)$ is given in terms of bit.

y	Nodes per Segment	P_s (before VT)	$\hat{P}_s$ (after VT)	$\hat{P}_d$ (after VT)	$\mathrm{I}(X^v; W^*)$ (in bits)	Effective Secret Bits	Comparison against RS
4.95	4	3×10^{-6}	3.3×10^{-12}	1.1×10^{-10}	≤ 256	≈ 33.3	
4.95	8	6×10^{-6}	1.6×10^{-11}	2.5×10^{-10}	≤ 144	≈ 145.3	
4.95	16	1.2×10^{-5}	6.6×10^{-11}	5.3×10^{-10}	≤ 80	≈ 209.3	
4.95	32	2.4×10^{-5}	2.7×10^{-10}	1.1×10^{-9}	≤ 44	≈ 245.3	
4.95	64	4.7×10^{-5}	1.1×10^{-9}	2.2×10^{-9}	≤ 24	≈ 265.3	
4.95	128	9.5×10^{-5}	4.5×10^{-9}	4.5×10^{-9}	≤ 13	≈ 276.3	$\leftarrow$
4.24	4	8.8×10^{-5}	2.9×10^{-9}	9.4×10^{-8}	≤ 256	≈ 59.8	
4.24	8	1.8×10^{-4}	1.4×10^{-8}	2×10^{-7}	≤ 144	≈ 171.8	
4.24	16	3.5×10^{-4}	5.9×10^{-8}	5×10^{-7}	≤ 80	≈ 235.8	
4.24	32	7.1×10^{-4}	2×10^{-7}	1×10^{-6}	≤ 44	≈ 271.8	

From the comparison of Tables 4 and 5, we observe a sufficient reliability for both approaches. The VT-like entries with a lower numbers of effective bits have been added for explanatory reasons. In terms of effective secret bits, the VT-like code outperforms the RS code by over 40%. Moreover, its expected implementation is simplified as no operations in Galois fields are required. Furthermore, instead of the burst error-correction by the RS code, the VT-like code maintains a better tamper-sensitivity since mostly adjacent intervals are corrected. Another advantage is that its bit mapping introduces less bias and therefore leaks less information.

Considering Table 5 more closely, we observe that for smaller segments with less nodes, a better reliability is achieved. However, at the same time more information is leaked by the helper data. In the simulation of 1.2×10^7 devices, no device failed, which gives a confident error rate $\ll 1 \times 10^{-6}$.

6 Conclusion

The majority of previous fuzzy extractor schemes is limited to binary PUF outputs and therefore impractical to use for higher-order alphabets. Moreover, the few existing works considering higher-order alphabets are limited to fixed-length bit mappings and equiprobable quantization.

This work introduces a variable-length bit mapping and a corresponding error correction scheme for an equidistant quantization. Its impact is manifold: it relieves designers of PUF systems of previously existing constraints regarding the selection of the quantization scheme, it results in a more efficient scheme and therefore a longer effective secret bit output, and also improves other desired properties such as tamper-sensitivity.

For the practical scenario considered, we are able to increase the number of effective secret bits by 40% while at the same time not requiring complex finite field operations as it would be the case for an RS decoder. While the results are already promising, we consider this only as a first step towards a more efficient use of higher-order alphabet PUFs.

Acknowledgements

The authors from Fraunhofer AISEC have been supported by the Fraunhofer Internal Programs under Grant No. MAVO 828 432. A. Lenz and A. Wachter-Zeh have been supported by the Technical University of Munich–Institute for Advanced Study, funded by the German Excellence Initiative and European Union Seventh Framework Programme under Grant Agreement No. 291763. Many thanks to Aysun Önalan for preparing the numbers of the RS-based fuzzy commitment scheme.

References

1. G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *ACM/IEEE Design Automation Conference (DAC)*, 2007.
2. J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Workshop on Cryptographic Hardware and Embedded Systems CHES 2007*, 2007.
3. P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2006.
4. V. Immler, M. Hennig, L. Kürzinger, and G. Sigl, "Practical aspects of quantization and tamper-sensitivity for physically obfuscated keys," in *Workshop on Cryptography and Security in Computing Systems (CS2)*, 2016.
5. A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *ACM Conference on Computer and Communications Security (CCS)*, 1999.
6. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology (EUROCRYPT)*, 2004.
7. C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, E. Oswald and P. Rohatgi, Eds., 2008.
8. M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, no. 1, 2010.
9. R. Maes, "Physically unclonable functions: Constructions, properties and applications," Dissertation, 2012.
10. M. Hiller, D. Merli, F. Stumpf, and G. Sigl, "Complementary IBS: Application specific error correction for PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 1–6.
11. S. Puchinger, S. Müelich, M. Bossert, M. Hiller, and G. Sigl, "On error correction for physical unclonable functions," in *International ITG Conference on Systems, Communications and Coding (SCC)*, Feb. 2015.
12. M. Hiller, M. Yu, and G. Sigl, "Cherry-picking reliable PUF bits with differential sequence coding," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2065–2076, 2016.
13. S. Puchinger, S. Müelich, M. Bossert, and A. Wachter-Zeh, "Timing attack resilient decoding algorithms for physical unclonable functions," in *International ITG Conference on Systems, Communications and Coding (SCC)*, Feb. 2017.
14. G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 766–769, 1984.
15. R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors (in russian)," *Automatika i Telemekhanika*, 1965.
16. V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals (in russian)," *Doklady Akademii Nauk SSR*, vol. 163, no. 4, pp. 845–848, 1965.
17. O. Günlü and O. Iscan, "DCT based ring oscillator physical unclonable functions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 8248–8251.
18. T. Stanko, F. N. Andini, and B. Skoric, "Optimized quantization in zero leakage helper data systems," *IEEE Transactions on Information Forensics and Security*, 2017.

19. J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M. Yu, "Efficient fuzzy extraction of PUF-induced secrets: Theory and applications," in *Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016.
20. F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann, "A formalization of the security features of physical functions," in *IEEE Symposium on Security and Privacy (S&P)*, 2011, pp. 397–412.
21. B. Colombier, L. Bossuet, V. Fischer, and D. Hely, "Key reconciliation protocols for error correction of silicon PUF responses," *IEEE Transactions on Information Forensics and Security*, 2017.
22. M. Hiller, M.-D. M. Yu, and M. Pehl, "Systematic Low Leakage Coding for Physical Unclonable Functions," in *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2015.
23. T. Ignatenko and F. M. Willems, "Information Leakage in Fuzzy Commitment Schemes," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 337–348, June 2010.
24. R. Maes, V. van der Leest, E. van der Sluis, and F. Willems, "Secure key generation from biased PUFs: extended version," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 121–137, 2016.
25. J. von Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, 1951.
26. M. Suzuki, R. Ueno, N. Homma, and T. Aoki, "Multiple-valued debiasing for physically unclonable functions and its application to fuzzy extractors," in *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, 2017.
27. M. Yu, M. Hiller, and S. Devadas, "Maximum likelihood decoding of device-specific multi-bit symbols for reliable key generation," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2015, pp. 38–43.
28. J. Delvaux and I. Verbauwhede, "Key-recovery attacks on various RO PUF constructions via helper data manipulation," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014.
29. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady*, 1966.
30. N. J. A. Sloane, "On single-deletion-correcting codes," in *Codes and Designs.* de Gruyter, 2002, pp. 273–292.
31. K. Saowapa, H. Kaneko, and E. Fujiwara, "Systematic deletion/insertion error correcting codes with random error correction capability," in *Defect and Fault Tolerance in VLSI Systems*, 1999.
32. G. I. Davida, Y. Frankel, and B. J. Matt, "On enabling secure applications through off-line biometric identification," in *IEEE Symposium on Security and Privacy (S&P)*, 1998, pp. 148–157.
33. F. Gray, "Pulse code communication," 1953, US Patent 2,632,058.
34. J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M. Yu, "Secure sketch metamorphosis: Tight unified bounds," *IACR eprint archive*, 2015.
35. F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes.* North-Holland, 1977.