Entropy Loss in PUF-based Key Generation Schemes: The Repetition Code Pitfall

Patrick Koeberl, Jiangtao Li, Anand Rajan and Wei Wu
Intel Labs
{patrick.koeberl, anand.rajan, jiangtao.li, wei.a.wu}@intel.com

***Abstract*—One of the promising usages of Physically Unclonable Functions (PUFs) is to generate cryptographic keys from PUFs for secure storage of key material. This usage has attractive properties such as physical unclonability and enhanced resistance against hardware attacks. In order to extract a reliable cryptographic key from a noisy PUF response a fuzzy extractor is used to convert non-uniform random PUF responses into nearly uniform randomness. Bösch et al. in 2008 proposed a fuzzy extractor suitable for efficient hardware implementation using two-stage concatenated codes, where the inner stage is a conventional error correcting code and the outer stage is a repetition code. In this paper we show that the combination of PUFs with repetition code approaches is not without risk and must be approached carefully. For example, PUFs with min-entropy lower than 66% may yield zero leftover entropy in the generated key for some repetition code configurations. In addition, we find that many of the fuzzy extractor designs in the literature are too optimistic with respect to entropy estimation. For high security applications, we recommend a conservative estimation of entropy loss based on the theoretical work of fuzzy extractors and present parameters for generating 128-bit keys from memory based PUFs.**

***Index Terms*—fuzzy extractor, entropy loss analysis, physically unclonable functions, secure key generation, system design**

I. Introduction

Most if not all cryptographic schemes require a key and are underpinned by the requirement to store this material securely. Conventional secure key storage schemes use non-volatile memory (NVM) technologies to store cryptographic secrets which can be vulnerable to attacks such as readout and cloning. Key generation and storage solutions based on Physically Unclonable Functions (PUFs) are emerging which promise high levels of assurance at low cost.

A PUF can be informally described as a physical system which when measured or challenged provides unique, repeatable and unpredictable responses. Creating a physical copy of the PUF with an identical challenge-response behavior is hard, thus resulting in a structure which is unclonable even by the manufacturer. Pappu introduced the PUF concept in [19]. In 2002, Gassend et al. introduced silicon PUFs in [7]. Silicon PUFs exploit the uncontrollable manufacturing variations which are a result of the IC fabrication process.

Since the first development of silicon PUFs, many silicon PUF constructions have been proposed. Lee et al. [13] proposed the first Arbiter PUF in 2004. Guajardo et al. [8] proposed the SRAM PUF in 2007. In 2008 Kumar et al. [12] introduced Butterfly PUFs and Maes et al. [16] proposed D Flip-Flop (D-FF) PUFs. SRAM PUFs, Butterfly PUFs, and D-FF PUFs are all memory based PUFs.

A promising usage of PUF is secure key generation in silicon [14], [21], [8], [15]. Using PUFs for key generation eliminates the need for storing keys in NVM technologies. In [21], [8], a cryptographic key-generation scheme is presented. It was stated in [17] that PUF based key generation provides advantages like physical unclonability and tamper evidence, compared to storing the keys in NVM.

Since PUF responses are inherently noisy and may not be uniformly random, a post-processing function is needed to condition the raw PUF responses into high-quality cryptographic keys. This post-processing function is known as the fuzzy extractor or helper data algorithm in the literature. Bösch et al. [3] proposed a fuzzy extractor scheme suitable for implementation using two-stage concatenated codes, where the inner stage is a conventional error correcting code such as BCH, Reed-Muller, or Golay code, and the outer stage is a repetition code. This fuzzy extractor offers simple encoding/decoding and efficient hardware implementation. Recent soft decision fuzzy extractors [17], [23] are also based on concatenated codes with the outer stage being a repetition code.

In this paper, we show that fuzzy extractors based on repetition codes are not without risk, particularly when the PUF response has low entropy. If the min-entropy of PUF is lower than 66%, then repetition code based fuzzy extractors could result in zero leftover entropy and should be avoided. Even if the PUF response has excellent entropy, e.g., with 95% min-entropy, we need a larger PUF to extract a 128-bit key than estimated in [3]. Recent entropy analysis on memory based PUFs in [22] shows that, on a particular implementation, the latch PUF has a min-entropy of 39%, the D-FF PUF has a min-entropy of 50% and the SRAM PUF has the largest min-entropy with 87%. Although most of the SRAM PUFs have good entropy, certain SRAM devices have been reported with low entropy [10]. Some memory-based PUFs exhibit low entropy and strong bias, e.g., D-FF PUFs [16].

A. Related Work

Dodis et al. [6], [5] provided a formal definition of fuzzy extractors that converts noisy data into cryptographic keys. This paper provides two fuzzy extractor constructions for the Hamming distance metric: code-offset construction and syndrome construction. The code-offset construction is frequently used in the PUF literature, e.g. in [3], [17], [23]. The paper also

978-1-4799-4112-4/14/$31.00

provides a formal treatment and analysis of entropy loss in the fuzzy extractor. The entropy analysis in our paper is primarily based on this work. Boyen further discussed the secure use of fuzzy extractors in [4].

Bösch et al. [3] presented the first efficient hardware implementation of a fuzzy extractor using concatenated codes where one of the codes is a repetition code. We refer to such approaches as repetition code based fuzzy extractors in this paper. Maes et al. [17] proposed a fuzzy extractor using soft decision error correction for memory-based PUFs. In soft decision fuzzy extractors, the error probability of each PUF cell is stored in the helper data, so that better decisions can be made during error correction. It has been shown in [17] that soft decision fuzzy extractors could reduce the PUF size by up to 58% when generating a 128-bit cryptographic key. This soft decision fuzzy extractor has limitations in that it requires multiple PUF measurements during setup and requires a large quantity of helper data. Recently Van der Leest et al. [23] proposed a soft decision fuzzy extractor using a single enrollment. The encoding scheme is the same as in [3] but the decoding uses soft decision decoding. It is not as efficient as [17] but better than the hard decision fuzzy extractor in [3]. All three fuzzy extractors [3], [17], [23] use repetition codes. Yu and Devadas [24] provided a different technique called index based syndrome to reduce entropy loss in the fuzzy extractor. This approach assumes the PUF output is a real value instead of a single bit and is applicable to some PUF designs such as the Ring Oscillator PUF [24]. However, this technique does not apply to memory-based PUFs which have binary outputs.

B. Our Contributions

We summarize our contributions of this paper as follows:

1) Although repetition code based fuzzy extractors [3], [17], [23] have been shown to have efficient hardware implementations, we discover that repetition code based fuzzy extractors suffer high entropy loss, particularly, when the PUF exhibits low min-entropy. We show that if the PUF min-entropy is lower than 66%, then a repetition code based fuzzy extractor could result in zero leftover entropy and should be avoided. We provide two concrete examples to demonstrate that the entropy loss is real. Furthermore, we show that it is likely impossible to construct any fuzzy extractors for low min-entropy and high error-rate PUFs.
2) We find that many of the fuzzy extractor designs [8], [3], [17], [23] in PUF-based key generation systems are too optimistic in entropy estimation, e.g., making an unrealistic assumption that PUFs have full min-entropy. In addition, the entropy loss due to the leftover hash lemma [9] in these schemes is neglected. As a result, these schemes may not have any leftover entropy in the extracted key for any practical constructions of memory-based PUFs. For high security applications, we recommend a conservative estimation of entropy loss based on the theoretical work of fuzzy extractors [6] and suggest parameters for generating 128-bit keys from memory-based PUFs.

C. Organization of the Paper

The rest of this paper is organized as follows. We first review the theory behind fuzzy extractors and then review the repetition code based fuzzy extractors in Section II. In Section III, we show that the repetition code based fuzzy extractors may suffer high entropy loss and could even be unusable when the raw PUF data has low min-entropy. In Section IV, we present a set of fuzzy extractor parameters for high assurance applications with a conservative estimation on entropy loss. We conclude our paper in Section V.

II. Review of Fuzzy Extractor

In this section, we begin with a description of our notations and some preliminaries on measurements of randomness. Let w and w' be two equal-length strings, we use $\mathsf{dist}(w, w')$ to denote the Hamming distance between them. Let $\mathcal{C}[n, k, d]$ be an error correcting code, where n is the length of the code, k is the size of message (if the code is linear, k is also called the dimension of the code), and d is the minimum distance of the code (which is the minimum Hamming distance between any two codewords). We use t to denote the maximum number of errors that $\mathcal{C}[n, k, d]$ can correct, where $d \geq 2t+1$. Let A and B be two probability distributions from a set $\mathcal{S}$. We define the *min-entropy* of A as

$$\mathbf{H}_\infty(A) = -\log(\max_{s\in\mathcal{S}} \Pr[A = s])$$

The *average min-entropy* of A given B is defined as

$$\tilde{\mathbf{H}}_\infty(A|B) = -\log(\mathbb{E}_{b\leftarrow B}[2^{-\mathbf{H}_\infty(A|B=b)}])$$

We define the *statistical distance* between A and B as

$$\mathbf{SD}(A, B) = \frac{1}{2}\sum_{s\in\mathcal{S}} |\Pr[A = s] - \Pr[B = s]|$$

From this point on, when we state an n-bit PUF response w has 95% min-entropy, we mean the min-entropy in w is $0.95n$.

A. Formal Definition of Fuzzy Extractor

Given a noisy and partially random input, such as a PUF response, a fuzzy extractor is an algorithm that extracts cryptographic keys from the noisy input. We use the formal definition of fuzzy extractors from [5] as follows.

Definition 1 (Fuzzy Extractor): An $(n, m, \ell, t, \epsilon)$-fuzzy extractor is a pair of randomized procedures: *generation (*Gen*)* and *reproduction (*Rep*)*, with the following properties:

1) The generation procedure on input $w \in \{0,1\}^n$ outputs an extracted key $k \in \{0,1\}^\ell$ and a helper data string $h \in \{0,1\}^*$, i.e., $(K, h) \leftarrow \mathsf{Gen}(w)$.
2) The reproduction procedure takes an element $w' \in \{0,1\}^n$ and a helper data string $h \in \{0,1\}^*$ as input. The correctness property of fuzzy extractors guarantees that if $\mathsf{dist}(w, w') \leq t$ and $(K, h) \leftarrow \mathsf{Gen}(w)$, then $\mathsf{Rep}(w', h) = K$.

3) The security property guarantees that for any distribution w on $\{0,1\}^n$ of min-entropy m, the string K is nearly uniform even if the adversary observes h. More specifically, if $(K,h) \leftarrow \mathsf{Gen}(w)$, then $\mathbf{SD}((K,h),(U_\ell,h)) \leq \epsilon$ where U_ℓ is the uniform distribution on $\{0,1\}^\ell$.

The security property states that the extracted key K is very close to a randomly chosen ℓ-bit key, with only a negligible statistical difference. To extract a 128-bit cryptographic key from a fuzzy extractor with 128-bit security, one can choose $\ell = 128$ and $\epsilon = 2^{-128}$ as parameters. The choices of n, m, and t will depend on the min-entropy and error rate of the PUF response. We shall discuss how to choose these parameters in Sections IV for high security applications.

B. Fuzzy Extractor based on Code-Offset Construction

Two fuzzy extractor constructions have been provided for the Hamming distance metric [5]: the code-offset construction and the syndrome construction. Both constructions have similar efficiency. In this paper, we review the code-offset construction as follows. The flow of the code-offset based fuzzy extractor is depicted in Figure 1.

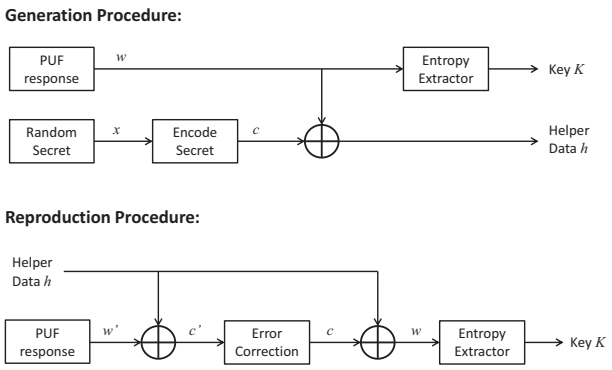

Fig. 1. Code-offset fuzzy extractor construction

1) **Generation procedure**: Let $\mathcal{C}[n,k,d]$ be an error correcting code. On input $w \in \{0,1\}^n$, this procedure chooses $x \in \{0,1\}^k$ randomly and computes $c = \mathcal{C}(x)$, the encoding of x. It computes the helper data as $h = w \oplus c$. The generation procedure also picks a universal hash function H and computes the key $K = H(w)$. It outputs (K,h).
2) **Reproduction procedure**: On input of an element $w' \in \{0,1\}^n$, a helper data string h, and a universal hash function H, this procedure first computes $c' = w' \oplus h$. It then decodes c' to get c, if $\mathsf{dist}(w,w') \leq t$. It reproduces $w = h \oplus c$ and outputs the key $K = H(w)$.

Note that the helper data usually includes the choice of the universal hash function H. For simplicity, we neglect H from the helper data. In many practical implementations of fuzzy extractors [3], [17], [23], universal hash functions are replaced with an LFSR-based Toeplitz hash [11] or other cryptographic functions (e.g., SHA or HMAC).

To understand the entropy loss of the code-offset construction, the following theorem has been proved in [5]. Let L be the entropy loss in the randomness extraction. For information theoretical security $L = 2\log(1/\epsilon) - 2$ due to the leftover hash lemma [9] and for computational security $L = \log(1/\epsilon)$ due to the generalized leftover hash lemma [2].

Theorem 1: [5] Given any $\mathcal{C}[n,k,d]$ code, the code-offset construction is an (n,m,ℓ,t,ϵ)-fuzzy extractor, where $\ell = m + k - n - L$. The entropy loss in the fuzzy extractor is $n - k + L$.

C. Review of Repetition Code based Fuzzy Extractors

Recently there have been many works [3], [17], [23] using concatenated codes to construct fuzzy extractors, more specifically, a repetition code combined with another Error Correcting Code (ECC). Various decoding algorithms have been discussed. The major goal is to reduce the size of the PUF response and the complexity for hardware implementation. Using a conventional ECC, such as a repetition code or a BCH code, the code-offset construction may be inefficient. As argued by Bösch et al. [3], using repetition code alone requires large PUF responses, using Golay code suffers high error probability, and BCH code has high decoding complexity. As a result, they proposed to use a concatenated code to address the PUF requirement and decoding complexity tradeoff.

In coding theory, concatenated codes are a class of ECC codes that achieve a longer codeword and larger code minimum distance by combining two shorter codes, i.e. an outer code $\mathcal{C}_1[n_1,k_1,d_1]$ and inner code $\mathcal{C}_2[n_2,k_2,d_2]$. The new code has $n = n_1 n_2$ and $d = d_1 d_2$. When used in a fuzzy extractor, intuitively, the outer code brings down the error rate, and the inner code eliminates the remaining errors. In Figure 2, we illustrate a concatenated code construction. The outer code is specified as a repetition code as in [3], [17], [23].

Codeword generation starts with l_2 blocks, each k_2-bit long. Each block is encoded by the inner code $\mathcal{C}_2$, the intermediate result has an identical number of blocks and each block is n_2-bit long. This inner code can be any variant of ECC, such as BCH, Reed-Muller, or Golay code. Lastly, every single bit is repeated n_1 times and concatenated to form the final codeword $\mathcal{C}_s$. The decoding procedure follows the reverse, decoding the repetition codeword first and the inner codeword afterwards.

III. Entropy Loss in Repetition Code based Fuzzy Extractors

In this section, we discuss the entropy loss in the repetition code based fuzzy extractors used in [3], [17], [23]. We begin with a short PUF response w and check the leftover entropy from the code-offset construction. Consider an n_1-bit subset of PUF response w and a repetition code $\mathcal{C}_1[n_1,1,n_1]$. In the code-offset construction, a secret bit x is chosen and $w \oplus \mathcal{C}_1(x)$ is output as helper data h. We wish to calculate the min-entropy in w or in x given the value of h. Clearly, the leftover entropy is at most 1 bit. Theorem 1 says that the leftover entropy in w is $m + 1 - n_1$, where m is the min-entropy in w.

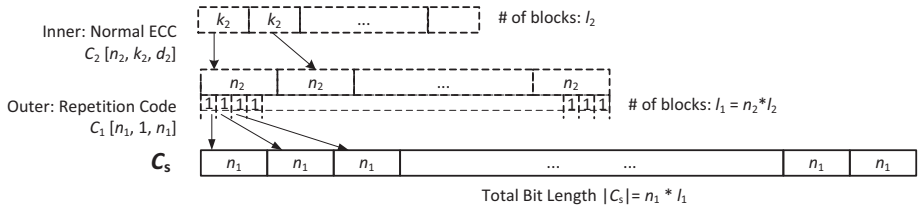

Fig. 2. Encoding flow for a concatenated code

$n_1 =$	3	5	7	9	11	13
$\mathbf{H}_\infty$= 95%	0.85	0.75	0.65	0.55	0.45	0.35
$\mathbf{H}_\infty$= 85%	0.55	0.25	0	0	0	0
$\mathbf{H}_\infty$= 75%	0.25	0	0	0	0	0
$\mathbf{H}_\infty$= 65%	0	0	0	0	0	0

TABLE I
LEFTOVER ENTROPY OF n_1-BIT PUF RESPONSE w FROM THE REPETITION CODE

w size	$\mathbf{H}_\infty(w)$	$\mathbf{H}(w)$	probability on guessing x
5	2.08	4.06	0.896
7	2.91	5.67	0.929
9	3.73	7.30	0.951

TABLE II
ENTROPY IN A BIASED PUF AND POTENTIAL HELPER DATA INFORMATION LEAKAGE

Table I summarizes the leftover entropy on various n_1 and various min-entropy. If the min-entropy in w is low, e.g., 65%, then there is zero leftover entropy left from the code-offset. Even if the min-entropy in w is high, e.g., 95%, the leftover entropy decreases quickly if we use a large repetition code. One may argue that the leftover entropy from Theorem 1 proved by Dodis et al. [5] is only a lower bound estimation. We give two concrete scenarios where entropy loss is a concern.

In the first scenario, suppose the first bit of the PUF response w is fixed (for example to 1) and the rest of the PUF bits are random. The entropy and min-entropy in w are both $n_1 - 1$. However, the leftover entropy on w given the helper data h is 0, because h leaks all the information about w and the secret bit x. An adversary can easily guess x by looking at the first bit of h. In practice, no study ever shows such fixed (or high probability) bits in memory based PUFs [8], [12], [16], [20]. Nevertheless, this scenario shows that a small drop in the raw PUF response entropy could result in zero leftover entropy. It matches with the entropy estimation in Theorem 1 as $m + 1 - n_1 = n_1 - 1 + 1 - n_1 = 0$.

In the second scenario, suppose the PUF response is strongly biased. For example, let the probability of 1 be 0.75 and the probability of 0 be 0.25. For a single PUF bit w, the min-entropy $\mathbf{H}_\infty(w) = -\log(0.75) = 0.415$ and the entropy $\mathbf{H}(w) = -0.75\log(0.75) - 0.25\log(0.25) = 0.811$. The (min-)entropy on a multiple bit w can be computed similarly. Table II shows that the min-entropy is about 40% and entropy is about 80%. Given the helper data $h = w \oplus \mathcal{C}_1(x)$, an adversary can guess the value of x as follows: she guesses $x = 0$ if the major bit in h is 1 and $x = 1$ otherwise. The adversary can guess x successfully with high probability (almost 0.9 probability for $n_1 = 5$ and better for larger n_1).

Even if the PUF response is slightly biased with good entropy (e.g., probability of being 1 is 0.6 with min-entropy 74% and entropy 97%), a repetition code of size 13 would give the adversary 0.9 probability of guessing x successfully. This scenario shows that even if the PUF response has good entropy (say 80% or even 97%), severe entropy loss is possible in a repetition-code based fuzzy extractor. In many real PUF implementations, both good and bad entropies have been observed. For example, Berg in his thesis [22] shows that the latch PUF has a low min-entropy of 39%, the D-FF PUF has a min-entropy of 50%, and the SRAM PUF has a min-entropy of 87%. A low entropy SRAM PUF has been reported in [10].

We now look at the concatenated-code based fuzzy extractor where the outer code is a repetition code. Let $\mathcal{C}_s$ be a concatenated code of $\mathcal{C}_1[n_1, 1, n_1]$ and $\mathcal{C}_2[n_2, k_2, d_2]$ with l_2 rounds. Essentially, $\mathcal{C}_s$ is an error correcting code $[l_2 n_2 n_1, l_2 k_2, d_2 n_1]$. Let the PUF response be w with m-bit min-entropy. The leftover entropy on w given the helper data h is $m + l_2 k_2 - l_2 n_2 n_1$ based on Theorem 1. If the PUF response has full entropy and is uniformly random, i.e., $m = l_2 n_2 n_1$, then the leftover entropy is $l_2 k_2$.

Table III shows repetition-code based fuzzy extractor methods from [3], [17], [18], [23] with required PUF size, failure rate, and leftover entropy for deriving a 128-bit key, assuming the PUF response has 100% min-entropy, 95% min-entropy, and 75% min-entropy, respectively. The methods from [3] are hard decision fuzzy extractors and the methods from [17], [18], [23] are soft decision fuzzy extractors. We only select methods with failure rate $\leq 10^{-6}$ for most of the cases. Observe that if the PUF response is random, then there is at least 168-bit leftover entropy. However, if the PUF response has 95% min-entropy, half of the methods have 0 leftover entropy and none of them have more than 128-bit entropy. If the PUF response has 75% min-entropy, no entropy remains.

We can see from Table III that there is not enough entropy left to extract a 128-bit cryptographic key if the min-entropy of w is 95% or 75%. This contradicts the claims from [3], [17], [23]. The main reason is that [3], [17], [23] are too optimistic in entropy estimation: they assume that the PUF response is uniformly random during the secure sketch procedure, and then consider entropy loss only during the randomness extraction process (termed privacy amplification in these papers). The fact that there is additional entropy loss in the secure

Ref.	$\mathcal{C}_1$	$\mathcal{C}_2$	l_2	PUF size	P_{Fail}	Leftover Entropy		
						100% $\mathbf{H}_\infty$	95% $\mathbf{H}_\infty$	75% $\mathbf{H}_\infty$
[3]	$R[3,1,3]$	$BCH[63,7,31]$	25	4725	2.0 E-5	175	0	0
[3]	$R[5,1,5]$	$BCH[226,86,43]$	2	2260	4.6 E-7	172	59	0
[3]	$R[11,1,11]$	$G[24,12,8]$	14	3696	7.1 E-6	168	0	0
[3]	$R[13,1,13]$	$RM[16,5,8]$	35	7280	1.6 E-7	175	0	0
[17]	$R[3,1,3]$	$RM[64,22,16]$	8	1536	1 E-8	176	99	0
[18]	$R[4,1,4]$	$RM[32,16,8]$	11	1408	6.5 E-7	176	105	0
[18]	$R[7,1,7]$	$RM[16,11,4]$	16	1792	1 E-8	176	86	0
[23]	$R[7,1,7]$	$RM[16,5,8]$	35	3920	3.7 E-7	175	0	0
[23]	$R[14,1,14]$	$RM[8,4,4]$	43	4816	3.3 E-7	172	0	0
[23]	$R[8,1,8]$	$G[24,12,8]$	15	2880	4.8 E-7	180	36	0

TABLE III

LEFTOVER ENTROPY ON THE PUF RESPONSE GIVEN THE HELPER DATA WITH VARIOUS FUZZY EXTRACTOR METHODS AND VARIOUS MIN-ENTROPY ON PUF. IN THE TABLE, R DENOTES REPETITION CODE, BCH DENOTES BCH CODE, RM DENOTES REED-MULLER CODE, G DENOTE GOLAY CODE, AND P_{Fail} DENOTES THE FAILURE RATE ON THE KEY.

sketch process was neglected in [3], [17], [23].

IV. PARAMETERS FOR HIGH SECURITY APPLICATIONS

In this section, we examine how to construct fuzzy extractors for memory-based PUFs in high security applications. As discussed earlier, many existing fuzzy extractor proposals [3], [17], [23] are over-optimistic on entropy estimation and do not have enough entropy for extracting a 128-bit key, even if the PUF min-entropy is 95%. These fuzzy extractors may be suitable for low-cost, resource constrained devices such as RFID or smart cards. But for high security or high assurance applications, we recommend a conservative estimate of entropy loss based on theories from Dodis et al. [5] so that we have high confidence about the leftover entropy in the extracted cryptographic key.

To be consistent with the literature and have a fair comparison, we consider the memory-based PUF error rate to be $p = 15\%$ and several possible min-entropy values for the PUF: 95%, 75%, and 50%. The goal is to extract a 128-bit cryptographic key K from PUF using fuzzy extractors with a failure rate $P_{\text{Fail}} < 10^{-6}$.

Let $\mathcal{C}_s[n,k,d]$ be a concatenated code of $\mathcal{C}_1[n_1,1,n_1]$ and $\mathcal{C}_2[n_2,k_2,d_2]$ with l_2 rounds. Based on Theorem 1, we need to make sure $\ell \leq m+k-n-L = m+l_2k_2-l_2n_2n_1-L$, where $\ell = 128$ is the key length, m is the min-entropy in the raw PUF response, and L is the entropy loss in randomness extraction. For 128-bit computational security, we can set $L = 128$ based on the generalized LHL in [2]. In other words, to extract a 128-bit key from PUF, we need to have $m + k - n$ (the leftover entropy of the PUF response given the helper data h) greater or equal to 256 bit.

We now show how we estimate the failure rate for a hard decision fuzzy extractor. Let us define a binomial cumulative distribution function: $\mathsf{binocdf}(t,n,p) = \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}$. The error rate for the repetition code $\mathcal{C}_1[n_1,1,n_1 = 2t_1+1]$ is defined as $P_1 = 1 - \mathsf{binocdf}(t_1,n_1,p)$. The error rate for the inner code $\mathcal{C}_2[n_2,k_2,d_2 = 2t_2+1]$ is defined as $P_2 = 1-\mathsf{binocdf}(t_2,n_2,P_1)$. The overall failure rate P_{Fail} of a hard decision fuzzy extractor is defined as $P_{\text{Fail}} = 1-(1-P_2)^{l_2}$.

Table IV shows a few possible fuzzy extractor parameters for 95% min-entropy PUFs in order to extract a 128-bit key with failure rate $< 10^{-6}$. Using a hard decision fuzzy extractor with a repetition code, we need a 4910-bit PUF to extract the key. Repetition code $R[5,1,5]$ seems to be most efficient among all possible repetition codes. Using the soft decision fuzzy extractor from [17], [18], we need a 3456-bit PUF to extract the key. Using the soft decision fuzzy extractor that requires only a single enrollment from [23], we need a 20544-bit PUF. Note that the choices of soft decision fuzzy extractors come from Table III with recalculated failure rate based on expanded PUF sizes in order to have sufficient entropy. Observe that if the min-entropy of the PUF is high enough, repetition code based fuzzy extractors are still effective. Also observe that soft decision fuzzy extractors are slightly more efficient than hard decision fuzzy extractors, with smaller ECC complexity and smaller PUF requirements. However, soft decision fuzzy extractors require multiple PUF measurements at the generation phase and require much larger helper data, which may be unacceptable for some applications.

We now look at min-entropies of 75% and 50%. We cannot find any possible constructions to extract a 128-bit key. We searched all possible BCH codes under 1023-bit length without any outcome. We believe that hard decision fuzzy extractors do not work for high error rate and low entropy PUFs. Soft decision fuzzy extractors with single enrollment [23] do not work either, since they heavily depend on a large repetition code. With low PUF entropy, a repetition code would result in zero leftover entropy in the key. The soft decision fuzzy extractor from [17] may work, but significant modification is needed since repetition codes cannot be used likely resulting in a larger and costly soft-decision maximum-likelihood decoder.

For PUFs with high error rate and low entropy, we propose using the dark bits method [1], i.e., run multiple PUF measurements during the generation procedure, and discard all the PUF bits that are noisy. Using this method, we are able to reduce the PUF error rate *without* incurring entropy loss on the remaining PUF bits. Thus we can extract cryptographic keys from these PUFs without using repetition codes in the

Method	$\mathcal{C}_1$	$\mathcal{C}_2$	l_2	PUF size	Entropy	P_{Fail}
hard FE	$R[3,1,3]$	$BCH[977,232,205]$	3	8793	256	1.9 E-7
hard FE	$R[5,1,5]$	$BCH[982,502,107]$	1	4910	256	8.2 E-7
hard FE	$R[5,1,5]$	$BCH[474,204,73]$	3	7110	256	3.1 E-8
hard FE	$R[7,1,7]$	$BCH[817,542,57]$	1	5719	256	4.9 E-7
hard FE	$R[7,1,7]$	$BCH[460,289,41]$	2	6440	256	7.1 E-7
soft FE[17]	$R[3,1,3]$	$RM[64,22,16]$	21	4032	260	2.6 E-8
soft FE[18]	$R[4,1,4]$	$RM[32,16,8]$	27	3456	259	1.6 E-6
soft FE[18]	$R[7,1,7]$	$RM[16,11,4]$	48	5376	259	3 E-8
soft FE[23]	$R[8,1,8]$	$G[24,12,8]$	107	20544	256	3.4 E-6

TABLE IV
FUZZY EXTRACTOR (FE) PARAMETERS TO EXTRACT A 128-BIT CRYPTOGRAPHIC KEY FOR HIGH SECURITY APPLICATIONS WITH FAILURE RATE $< 10^{-6}$, ASSUMING PUF ERROR RATE IS AT MOST 15% AND PUF MIN-ENTROPY IS AT LEAST 95%

first instance. Due to the space limit, we will report more details of our proposal in the full version of this paper.

V. CONCLUSION

In this paper, we show that, while repetition codes are heavily used in recently proposed fuzzy extractors due to their simplicity and rapid reduction of error rate, their use is risky when the min-entropy in the PUF response is low. We also show that parameters in the existing fuzzy extractor literature are over-optimistic in entropy estimation and may not be suitable for high security applications. We recommend a conservative estimation of entropy loss based on the theoretical work of fuzzy extractors. An open question is whether there is a secure way to extract cryptographic keys from low-entropy and high-error PUFs in applications where multiple PUF measurements in the generation procedure are not possible.

REFERENCES

[1] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In *Advances in Cryptology - ASIACRYPT 2009*, pages 685–702, Berlin, Heidelberg, 2009. Springer.

[2] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover hash lemma, revisited. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *LNCS*, pages 1–20. Springer, 2011.

[3] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In *CHES'08: Proceedings of 10th International Workshop on Cryptographic Hardware and Embedded Systems*, volume 5154 of *LNCS*, pages 181–197, 2008.

[4] X. Boyen. Reusable cryptographic fuzzy extractors. In *ACM Conference on Computer and Communications Security*, pages 82–91. ACM, 2004.

[5] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[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*, volume 3027 of *LNCS*, pages 523–540. Springer, 2004.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *ACM Conference on Computer and Communications Security*, pages 148–160, New York, NY, USA, 2002. ACM Press.

[8] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems Workshop*, volume 4727 of *LNCS*, pages 63–80, September 2007.

[9] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[10] P. Koeberl, J. Li, R. Maes, A. Rajan, C. Vishik, and M. Wójcik. Evaluation of a PUF device authentication scheme on a discrete 0.13um SRAM. In *Proceedings of 3rd International Conference on Trusted Systems (INTRUST)*, volume 7222 of *LNCS*, pages 271–288. Springer, 2011.

[11] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology - CRYPTO '94*, volume 839 of *LNCS*, pages 129–139. Springer, 1994.

[12] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls. The butterfly PUF: Protecting IP on every FPGA. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 67–70, June 2008.

[13] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication application. In *Proceedings of the Symposium on VLSI Circuits*, pages 176–159, 2004.

[14] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, October 2005.

[15] R. Maes, A. V. Herrewege, and I. Verbauwhede. PUFKY: A fully functional PUF-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems, CHES 2012*, pages 302–319. Springer, 2012.

[16] R. Maes, P. Tuyls, and I. Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, page 17, Eindhoven,NL, 2008.

[17] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2009, pages 332–347. Springer-Verlag, 2009.

[18] R. Maes, P. Tuyls, and I. Verbauwhede. Soft decision helper data algorithm for SRAM PUFs. In *ISIT'09: Proceedings of the 2009 IEEE international conference on Symposium on Information Theory*, pages 2101–2105, Piscataway, NJ, USA, 2009. IEEE Press.

[19] R. S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.

[20] P. Simons, E. van der Sluis, and V. van der Leest. Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, pages 7 –12, june 2012.

[21] E. Simpson and P. Schaumont. Offline hardware/software authentication for reconfigurable platforms. In *8th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *LNCS*, pages 311–323. Springer, 2006.

[22] R. van den Berg. Entropy analysis of physical unclonable functions. Master's thesis, Technical University of Eindhoven, August 2012.

[23] V. van der Leest, B. Preneel, and E. van der Sluis. Soft decision error correction for compact memory-based PUFs using a single enrollment. In *Cryptographic Hardware and Embedded Systems, CHES 2012*, volume 7428 of *LNCS*, pages 268–282. Springer Berlin Heidelberg, 2012.

[24] M.-D. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design Test of Computers*, 27(1):48–65, Jan.-Feb. 2010.