Estimating the Unpredictability of Multi-Bit Strong PUF Classes

Ahmed Bendary
Department of ECE
The Ohio State University
Columbus, OH 43210, USA
bendary.1@osu.edu

Wendson A. S. Barbosa
Department of Physics
The Ohio State University
Columbus, OH 43210, USA
desabarbosa.1@osu.edu

Andrew Pomerance
Potomac Research LLC
Alexandria, VA 22314, USA
andrew@potomacresear.ch

C. Emre Koksal
Department of ECE
The Ohio State University
Columbus, OH 43210, USA
koksal.2@osu.edu

***Abstract*—With the ongoing advances in machine learning (ML), cybersecurity solutions and security primitives are becoming increasingly vulnerable to successful attacks. Strong physically unclonable functions (PUFs) are a potential solution for providing high resistance to such attacks. In this paper, we propose a generalized attack model that leverages multiple chips jointly to minimize the cloning error. Our analysis shows that the entropy rate over different chips is a relevant measure to the new attack model as well as the multi-bit strong PUF classes. We explain the sources of randomness that affect unpredictability and its possible measures using models of state-of-the-art strong PUFs. Moreover, we utilize min-max entropy estimators to measure the unpredictability of multi-bit strong PUF classes for the first time in the PUF community. Finally, we provide experimental results for a multi-bit strong PUF class, the hybrid Boolean network PUF, showing its high unpredictability and resistance to ML attacks.**

1. Introduction

Ubiquitous connectivity and computing have supported all aspects of our lives. However, benefits are threatened by the escalation of cybersecurity attacks in recent years. Critical data breaches have affected businesses, health care, educational institutions, banks, and governments. Existing cybersecurity solutions are based on protected secret keys that may also be vulnerable to leakage. Recently, physically unclonable functions (PUFs) have emerged as a powerful cybersecurity solution that generates sequences on demand that are used as secret keys. With PUFs, generated keys are generated on demand and not stored, which reduces their susceptibility to compromise.

PUFs can be implemented on chips and designed to exploit random variations in the manufacturing process [1], [2]. They are small in size, low in cost and cannot be replicated even with the same manufacturing process, that is, "physically unclonable". Typical use cases are authentication [1], [2], [3], [4], secret sharing [2], and intellectual property protection [5], [6]. The input (denoted as a "challenge") to a PUF chip is mapped to the output (a "response") based on the design and the unknown physical parameters. PUFs are required to be "reliable" such that applying the same challenge many times reproduces the same response with a low probability of error. Meanwhile, even with the knowledge of many challenge-response pairs (CRPs), an attacker cannot predict unseen responses with a low probability of error, i.e., "unpredictable".

PUFs are categorized as weak or strong according to the size of the challenge-response space. A strong PUF has a challenge-response space that grows rapidly with the size of the PUF (the length of the challenge) such as the Arbiter PUF [1], the Ring Oscillator (RO) PUF [2], and the hybrid Boolean network (HBN) PUF [7]. Those PUFs can be categorized also as delay-based PUFs that depend on the timing information of the circuit. Weak PUFs are those that have a relatively small number of challenges, and examples are the static random access memory (SRAM) PUF [5], [8], and the flip-flop PUF [9]. Both the SRAM PUF and flip-flop PUF are memory-based PUFs that depend on the cell state when powered on. We focus on strong PUFs since the large challenge-response space cannot be exhausted easily even when the attacker had previous access to the PUF chip [10], [11]. In particular, the HBN-PUF generates multi-bit responses, unlike existing strong PUFs that produce only single-bit responses. We consider the HBN-PUF as our use case to illustrate our proposed attack model and unpredictability metric, rather than existing multi-bit strong PUFs [12], [13], [14], [15], [16]. We note that comparing existing multi-bit strong PUFs is beyond the scope of this paper and will be considered as a separate future work.

1.1. Related Work

Existing ML attacks [17], [18], [19], [20] can efficiently estimate the unknown parameters and fully characterize PUFs with a sufficient subset of CRPs. However, the CRPs are simulated. Even when hardware CRPs are used as in [21], [22], [23], only single-bit PUFs are considered. In addition, these modeling attacks do not use information from other chips in the class. New attack models are required to consider chips jointly as well as multi-bit strong PUFs. Parallel to our results of high entropy rate of HBN PUF, the authors in [24] shows high resistance of HBN class to a novel ML attack that outperforms existing attacks and uses many chips jointly as proposed in our work.

In the literature, many performance metrics, such as uniqueness, bit-aliasing, steadiness, diffuseness,..., etc., are proposed to assess and compare PUFs [9], [25], [26], [27], [28], [29], [30], [31]. Some of these metrics are useful from a design perspective [27], [30] so that weaknesses in the PUF design can be discovered and improved. However, achieving good scores on these metrics is necessary but not sufficient to guarantee the unpredictability of PUFs[1]. Other metrics are either qualitative [28], [29] or specific to a PUF design [25], [26], [31]. There are established provable methods and theoretically sound bounds established to assess the unpredictability of the PUFs; see, e.g., [32], [33]. However, all these methods are 1) suitable for single-bit PUFs, 2) do not consider the relation between different responses, nor consider many chips jointly. The authors in [32] assume independent and biased response bits, while the approach in [33] sums up relationships between existing metrics such as uniqueness in terms of Hamming distance (HD) and min-entropy.

One of the major challenges to PUFs is non-invasive attacks, in particular, machine learning (ML) attacks. An attacker collects CRPs from different PUF chips or eavesdrops on a target PUF and chooses a ML technique to predict the responses to unseen challenges. Most existing work proposes a design for a PUF and runs a handful of well-known ML attacks to establish a cloning error for those attacks (which is usually reported to be approximately 0.5). This is only a statement of the ML attacks the authors ran on their PUF and does not establish a bound for all ML attacks. Thus, a strong notion of the unpredictability of PUFs is still needed, where a demanding task is to verify whether a given PUF class is resilient to non-invasive attacks or not. That is, we are motivated to carefully assess the unpredictability of any given PUF class.

Many works in the PUF literature (see [34], [35], [36]) use the term "entropy rate" to refer to the normalized entropy per bit. Other works calculate the entropy rate of CRPs for each single chip separately [37] or estimate it using compression [9], [38]. In this paper, the entropy rate is estimated using many chips jointly to assess the unpredictability of a PUF class.

Lastly, estimating the entropy requires a number of samples that scales up with the cardinality of responses. Thus, existing PUF entropy calculations consider compression techniques [9], [38] to estimate an upper bound on the entropy, use the min-entropy [27], [28], [39], [40], [41] to estimate a lower bound, or even use plug-in estimators [42], [43]. Although the entropy estimation methods in [44] are practical, the estimators used in this paper are theoretically sound with proven guarantees for convergence and sample complexity. Considering such recent and efficient estimators could be a huge step toward assessing a PUF class unpredictability. We note that modeling PUFs mathematically or using an empirically trained model to accurately derive the entropy rate is beyond the focus of this manuscript. It is of great importance for future work, but it is not an easy task due to the increased complexity of newly designed PUFs.

1. We show that, reversely, high unpredictability measured by entropy rate guarantees high scores on these metrics, while low unpredictability does not imply which metric(s) is low.

1.2. Contributions

In this paper, we provide simple and unified mathematical models for state-of-the-art PUFs to explain sources of randomness that affect reliability and unpredictability. Unlike existing modeling attacks, we consider a generalized attack model that leverages additional information in other chips from the same class.

Many existing works have proposed heuristic metrics to assess the unpredictability of PUFs, and in particular the new aspects of the HBN-PUF stress existing approaches to assessing the unpredictability. We bridge this gap and propose an information-theoretic framework to tackle this problem and determine whether a PUF class is unpredictable, which can indicate whether it is resilient to machine learning attacks or not. Then, we discuss the existing uniqueness measures and illustrate their shortcomings with counterexamples. As a counter, we show that entropy is a sufficient, precise, and consistent metric to measure the uniqueness of strong PUFs.

Since PUF responses are sent via a public channel, unseen responses are vulnerable to prediction/cloning attacks. Thus, CRPs should be unpredictable in two different aspects. One aspect is across different PUF chips (Inter-PUF unpredictability), i.e., given a challenge, the response of a PUF chip is unpredictable from other PUF chips. The other is across different responses of a single PUF chip (Intra-PUF unpredictability); i.e., given a set of CRPs of a PUF chip, the remaining responses are unpredictable. We explain two related quantities, in terms of joint and conditional probability distributions of CRPs, and their relation to different attacks. In the absence of these distributions, we provide different finite-order entropy rate measures using the available samples. In addition, we explain how to detect the possible deterministic behavior of a single PUF chip and decide whether to exclude it or not.

Lastly, we provide experimental results using recent efficient entropy estimators, new to the PUF community, to evaluate the unpredictability of the HBN-PUF, showing its high entropy rate, and hence high resistance to ML attacks. The contributions of this paper can be summarized as follows:

- We propose a generalized attack model that considers the responses of multiple PUF chips jointly to improve the estimation of responses for unseen challenges.
- To evaluate the strength of a PUF class, we introduce entropy rate as a metric, which is suitable for the new attack model as well as for multi-bit strong PUF classes such as HBN PUF. Unlike existing works that consider single chips separately or denote the normalized entropy per bit as the entropy rate, we estimate the entropy rate across different chips and responses.

- Moreover, we use efficient entropy estimators, which are being used for the first time in the PUF community. We also collect a significant amount of real data of 256-bit HBN PUF and conduct extensive experimental analysis to estimate the entropy rate for the HBN PUF for the first time.

Organization. The paper is organized as follows. The PUF model, definitions, and other preliminaries are introduced in section 2. A new attack model is introduced and discussed in section 3. A measure for the new attack model, which is also suitable for a PUF class unpredictability is provided and investigated in subsection 3.2. In section 4, finite-order entropy rate estimation based on the available sample size is introduced. Then, measuring the uniqueness of different chips and testing a single PUF chip to detect possible deterministic behavior are provided in subsection 5.1 and subsection 5.2, respectively. Experimental results using efficient entropy estimators for HBN-PUF are given in section 6, and the paper is concluded in section 7.

2. PUF Model and Preliminary

First, we define a PUF class and a PUF chip, which is an important step in choosing the appropriate performance metrics. We focus primarily on the mapping function to understand sources of randomness and the basic parameters that affect reliability and unpredictability. Let $\mathbb{R}$ and $\mathbb{Z}^+$ be the set of real numbers and the set of positive integers, respectively, and $N, N_r, N_v,\ N_w \in \mathbb{Z}^+$. Furthermore, the continuous-time version of a binary input x is denoted by $x(t)$, which has a transition from 0 to 1 (or 1 to 0) in a time of rise (fall) denoted by τ_r (τ_f). The cardinality of a set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. Let the challenge set, $\mathcal{C} = \{0,1\}^N$, and the response set, $\mathcal{X} = \{0,1\}^{N_r}$, be metric spaces with a distance function, e.g. HD d_H.

Definition 1 (A PUF Class)**.** *A class of PUFs is a discrete stochastic process of a finite number of random variables indexed by the challenge space and defined by a deterministic mapping (determined by the design/structure) as follows:*

$$f : \mathcal{C} \times \mathcal{V} \times \mathcal{W} \to \mathcal{X}, \tag{1}$$

where the set $\mathcal{V} \subset \mathbb{R}^{N_v}$ and the set $\mathcal{W} \subset \mathbb{R}^{N_w}$ are respectively the sets of random parameters that result from 1) variations/imperfections in the manufacturing process, e.g. path delay, and 2) model errors, e.g. thermal noise and logic gates' error propagation [45], and affect the unpredictability and reliability of the PUF class. The former set of parameters leads to variations that are non-ergodic across different observations of the response for the same challenge. These factors constitute the main and desired source of variations across distinct PUFs within the same class. The latter factors lead to variations for a given PUF across responses for the same challenge. Hence, they lead to unreliable operation if not properly addressed.

Definition 2 (A PUF chip)**.** *A PUF chip is a mapping, $f : \mathcal{C} \times \mathcal{V}_m \times \mathcal{W} \to \mathcal{X}$, where $\mathcal{V}_m \subset \mathcal{V}$ is a set of unknown deterministic parameters (generated by the manufacturing process). Moreover, the response at time t after applying the k^{th} challenge to the m^{th} chip is given by:*

$$\mathbf{X}_{k,m} = f\left(\mathbf{c}_k, \mathbf{v}_m, \boldsymbol{W}\right),\ 1 \le k \le 2^N, \tag{2}$$

where the random vector $\mathbf{X}_{k,m} = \left[x^n_{k,m}\right] \in \mathcal{X}$, $\mathbf{v}_m = [v^m_i] \in \mathcal{V}_m$ is an unknown realization of the random vector $\boldsymbol{V}$ resulted from the manufacturing process of the m^{th} chip, and $m \in \mathbb{Z}^+$.

$\mathbf{C} \in \{0,1\}^N$
Challenge
PUF
$\mathbf{X} \in \{0,1\}^{N_r}$
Response

Figure 1. A PUF maps a challenge to a response.

It should be clear that, when the set of parameters from the manufacturing process is deterministic, a PUF chip is a noisy realization of the PUF class. We provide unified continuous-time models for state-of-the-art strong PUFs in AppendixA. We show that different PUF designs are based on the similar overall idea that there are nonergodic variations due to $\mathbf{V}$ that affect the unpredictability and ergodic noise due to $\mathcal{W}$ that affect the reliability. The response after applying the k^{th} challenge is given by:

$$\mathbf{X}_k = f\left(\mathbf{c}_k, \boldsymbol{V}, \boldsymbol{W}\right) \tag{3}$$

where the random vectors $\mathbf{X}_k = [x^n_k] \in \mathcal{X}$, $\mathbf{c}_k = [c_{k,n}] \in \mathbb{R}^N$, $\boldsymbol{V} = [v_i] \in \mathcal{V}$, and $\mathbf{W} = [w_i] \in \mathcal{W}$.

Remark 1. *The unpredictability of a PUF class is inherited from the random variations of the manufacturing process as well as the mapping (the PUF structure/design). On the other hand, the reliability of PUFs is affected by thermal noise and the mapping itself. Although other parameters can change the unpredictability and reliability, such as the supply voltage, operating temperature, aging, etc., we focus mainly on the mapping, the random variations of the manufacturing process, and the thermal noise. In addition, we restrict our representation of the response to binary sequences $\in \mathcal{X}$ and assume that any necessary digitization is part of the physical process, which is more relevant to cryptographic applications. In the case of HBN-PUF, the measurement time is a design parameter. Later, wherever relevant, we choose an optimal measurement time that maximizes both reliability and unpredictability. In this work, we focus on the unpredictability and assume that such parameters affect the unreliability of PUFs since the enrollment phase is done under specific conditions, and any changes are considered a source of unreliability. Related entropy rate measures should be investigated in future work.*

Definition 3. *(Minimum Cloning Error) For a stochastic process $\mathcal{X}$ with entropy rate $H(\mathcal{X})$, let the binary entropy $H(p_{\mathcal{X}}) \triangleq H(\mathcal{X})/N_r$ per bit. An attacker might be able to achieve an average bit prediction error of the stochastic process output, denoted as cloning error p_{cl}, that is no less the probability of error for the binary entropy $p_{\mathcal{X}}$ where $0 \le p_{\mathcal{X}} \le p_{cl} \le 0.5$.*

Typical use cases of PUFs are authentication, secret sharing, and intellectual property protection. A common phase for utilizing PUFs is the enrollment phase where a trusted party (Alice) queries the PUF and stores different CRPs in a look-up table. Then Alice shares the PUF with a legitimate user (Bob). In Table 1, we give examples of CRPs of different 5-bit numerical toy PUFs.

For secret sharing, for example, the second phase is utilization. In this phase, a challenge is sent to Bob via a public channel, who queries the PUF and is required to reconstruct the same stored response corresponding to that challenge. However, the output response is perturbed by noise; i.e., the response is unreliable and requires careful reconstruction. Alice sends helper data (in addition to the challenge) for error correction, and Bob reconstructs the correct response, which is at Alice in a look-up table. This response should be unpredictable.

3. The Attack Model

We assume that noiseless CRPs are sent on a public channel similar to modeling attack assumptions in [11], [17], [18], [20]. Throughout this paper, we assume that Alice performs several queries for each challenge to correct noise and determine a ground truth for the corresponding response. Consequently, we ignore the noise and consider the PUF class mapping in (3) as a stochastic process and the mapping of a PUF chip in (2) as deterministic mapping. Given the k^{th} challenge, the corresponding response over all possible PUF chips is a random variable, $\mathbf{X}_k \in \mathcal{X}$.

To estimate the unknown parameters, $\mathbf{v}_m$, of a single-bit strong PUF, ML framework assigns, for example, a logistic model to the probability $p(\mathbf{x}_{k,m}, \mathbf{c}_k|\mathbf{v}_m)$. The authors in [20] successfully estimated the unknown parameters for different single-bit strong PUFs and predicted responses for unseen challenges with more than 99% prediction rate on simulated PUFs. Therein, the estimation problem is minimizing the negating log-likelihood as follows:

$$\hat{\mathbf{v}}_m = \underset{\mathbf{v}_m \in \mathcal{V}}{\arg\min} \sum_k -\ln p(\mathbf{x}_{k,m}, \mathbf{c}_k|\mathbf{v}_m), \tag{4}$$

where the logistic sigmoid is assigned as $p(\mathbf{x}_{k,m}, \mathbf{c}_k|\mathbf{v}_m) = (1 + e^{-\mathbf{x}_{k,m} f(\mathbf{c}_k, \mathbf{v}_m)})^{-1}$ and the unknown parameters are estimated via iterative optimization techniques. For the existing single-bit strong PUFs such as RO and Arbiter, the mapping function is as simple as an additive linear delay architecture. Thus, with a sufficient subset of CRPs, ML attacks [17], [18], [20] can efficiently estimate the unknown parameters to fully characterize the simulated PUF, and predict the remaining CRPs with high prediction rates.

Existing attack models measure the unpredictability of such single-bit strong PUFs with the entropy of a PUF [42], [43]. The entropy of a PUF is measured as follows:

$$H(PUF_m) \triangleq -\sum_k p(\mathbf{x}_{k,m}, \mathbf{c}_k) \log p(\mathbf{x}_{k,m}, \mathbf{c}_k). \tag{5}$$

However, this entropy is a measure of the complexity of the mapping function and the dimension of the unknown parameters rather than the uncertainty and randomness of a PUF chip. Also, it measures whether the mapping function spans a large space of CRPs or not. Other work in the literature [37] considers the entropy rate of a single PUF, i.e., the remaining entropy in the k^{th} response after observing $(k-1)$ CRPs, as a theoretical lower bound of the prediction rate as follows.

$$H(\mathbf{x}_{k,m}) \triangleq -\sum_k p(\mathbf{x}_{k,m}|\mathbf{x}_m^{k-1}) \log p(\mathbf{x}_{k,m}|\mathbf{x}_m^{k-1}), \tag{6}$$

where the sequence $\mathbf{x}_m^k \triangleq (\mathbf{x}_{1,m}, \dots \mathbf{x}_{k,m})$. In addition, up to the authors' knowledge, the existing works in the literature consider only a single PUF for modeling attacks or measuring performance metrics and do not consider many PUFs jointly to predict unseen CRPs. Utilizing other PUFs jointly may help to learn the mapping function as well as estimate common unknown parameters.

3.1. Generalized Attack Model

In our attack model, we consider multi-bit strong PUFs such as HBN PUF. We assume that the adversary is powerful who 1) owes (or observes CRPS from) many PUF chips from the same class, 2) jointly utilizes all seen CRPs from the intended chip as well as the corresponding responses from the owned chips, and 3) predicts a response for an unseen challenge as follows.

Definition 4 (Generalized Attack Model)**.** *An adversary designs an optimal predictor that minimizes the expected loss of predicting unseen responses from previously seen responses jointly with responses from other PUF chips as follows.*

$$\widehat{\mathbf{X}}_k = \underset{\mathbf{x}_k \in \mathcal{X}}{\arg\min} \, \mathbb{E}\left[\ell\left(\mathbf{X}_k, \mathbf{x}_k\right) | \mathbf{X}^{k-1} = \mathbf{x}^{k-1}, \mathbf{C}_k = \mathbf{c}_k\right], \tag{7}$$

where $\ell(\cdot)$ is a loss function, the expectation is taken over $\mathbb{P}_{\mathbf{X}_k|\mathbf{X}^{k-1}=\mathbf{x}^{k-1},\mathbf{C}_k=\mathbf{c}_k}$, $\mathbf{X}^k \triangleq (\mathbf{X}_1, \dots \mathbf{X}_k)$, and $\mathbb{P}_{\mathbf{X}^k|\mathbf{C}^k}$ is the joint distribution of the class responses. The well known minimum mean square estimate is given as $\widehat{\mathbf{X}}_k = \mathbb{E}\left[\mathbf{X}_k|\mathbf{X}^{k-1} = \mathbf{x}^{k-1}, \mathbf{C}_k = \mathbf{c}_k\right]$.

Example 1. *(Predicting a Response) In this example, we illustrate the prediction in* (7) *using Table 1. Suppose an attacker wants to attack PUF$_1$. The attacker already owes $M-1$ chips: PUF$_2$ to PUF$_M$, either by manufacturing or simulation. The attacker has observed responses $\mathbf{x}_{0,1}$ to $\mathbf{x}_{28,1}$ from PUF$_1$ and has the corresponding responses from all other chips PUF$_2$ to PUF$_M$, $\mathbf{x}_{0,m}$ to $\mathbf{x}_{28,m}$, $m \in \{2, \dots, M\}$. The predictor in* (7) *utilizes all known responses to the previously sent challenges from all chips to predict the response of PUF$_1$ for an unseen challenge.*

Remark 2. *Unlike attack models in the PUF literature, the prediction in* (7) *has two dimensions, one is across different chips and the second is across responses. Thus, our generalized attack model has the potential to extract more information about the PUF class and enhance the prediction than any other attack model.*

TABLE 1. Numerical toy PUF example: CRPs of different 5-bit PUFs.

Challenges	Responses	PUF_1	PUF_2	PUF_3	$\cdots$	PUF_m	$\cdots$	PUF_M	PUF_{M+1}	PUF_{M+2}
00000	$\mathbf{X}_0$	00000	00000	00000	$\cdots$	00000	$\cdots$	00000	00000	00000
00001	$\mathbf{X}_1$	01101	10111	00011	$\cdots$	00001	$\cdots$	00011	00001	00100
00010	$\mathbf{X}_2$	01011	11011	10111	$\cdots$	00001	$\cdots$	10111	00010	01000
00011	$\mathbf{X}_3$	00110	10100	10100	$\cdots$	00000	$\cdots$	10100	00011	01100
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
01111	$\mathbf{X}_{15}$	11000	11010	00010	$\cdots$	01010	$\cdots$	00010	01111	11001
10000	$\mathbf{X}_{16}$	00111	00101	11101	$\cdots$	10101	$\cdots$	11101	10000	11101
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
11100	$\mathbf{X}_{28}$	11001	01011	01011	$\cdots$	11111	$\cdots$	01011	11100	10011
11101	$\mathbf{X}_{29}$	10100	00100	01000	$\cdots$	11110	$\cdots$	01000	11101	10111
11110	$\mathbf{X}_{30}$	10010	10000	11100	$\cdots$	11110	$\cdots$	11100	11110	11011
11111	$\mathbf{X}_{31}$	11111	11111	11111	$\cdots$	11111	$\cdots$	11111	11111	11111
e.g.1: c_{k_1}	$\mathbf{X}_{k_1}$	10101	01010	01010	$\cdots$	10101	$\cdots$	01010	10101	10101
e.g.2: c_{k_2}	$\mathbf{X}_{k_2}$	11110	11110	11111	$\cdots$	11111	$\cdots$	11110	10101	11111

Theorem 1. *The generalized attack model that utilizes CRPs jointly from many PUFs has a minimum cloning error that is no greater than the minimum cloning error of any other attack model under the same conditions.*

Proof. The proof is simple, based on the fact that conditioning reduces entropy, which increases the prediction rate. By Definition 3, the minimum cloning error for an attacker that uses only PUF_m equals the probability p_m where $H(p_m) = H(PUF_m)/N_r$. Similarly, for the generalized attack model, the minimum cloning error equals the probability p'_m where $H(p'_m) = H(PUF_m|PUF_1, \cdots, PUF_{m-1})/N_r$. Since

$$H(PUF_m|PUF_1, \cdots, PUF_{m-1}) \leq H(PUF_m), \quad (8)$$

we have

$$p'_m \leq p_m. \quad (9)$$

□

The "same conditions" means that both attack models have the same sample size and methodology/technique, and differ only on the amount of information used. Throughout this paper, we omit the conditioning on challenges since it is obvious from the subscript of the response. Thus, $\mathbb{P}_{\mathbf{X}^k|\mathbf{C}^k}$ is denoted as $\mathbb{P}_{\mathbf{X}^k}$.

3.2. Unpredictability Measure for the Generalized Model

Since the mapping of a PUF class is a stochastic process, an attacker has nothing to do except guess responses based on the joint probability distribution. For k i.i.d. random responses (the members of the stochastic process), we need $kH(\mathbf{X})$ bits, where $H(\mathbf{X})$ is the entropy of a random response. However, if these responses are not independent, the joint entropy $H(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k)$ grows (asymptotically) linearly with k at a rate:

$$H(\mathcal{X}) = \lim_{k\to\infty} \frac{1}{k} H(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k), \quad (10)$$

which is the well-known entropy rate of the stochastic process [46]. Thus, the entropy rate suffices to measure the Inter-PUF unpredictability. We note that a related quantity, the partitioning entropy [26], describes the unpredictability/complexity of the dynamical systems and is related to the metric entropy (Kolmogorov-Sinai entropy).

Without loss of generality, for a finite number of challenges, we can calculate two quantities related to the entropy rate:

1) The normalized joint entropy $\frac{1}{k}H(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k)$, which measures the average Inter-PUF unpredictability contained in k responses excluding any dependencies.
2) Conditional entropy $H(\mathbf{X}_k|\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{k-1})$ measures how much Inter-PUF unpredictability left in the k^{th} response after observing $(k-1)$ responses.

The second quantity is relevant to an attack scenario in which the attacker observes several CRPs and tries to predict an unseen response. Thus, the entropy rate gives the maximum number of CRPs that can be used before an attacker can successfully predict all the remaining CRPs and hence, clone the PUF. In other words, it measures the maximum number of independent CRPs that can be used.

On the other hand, the average entropy assumes that responses are independent and does not measure dependencies between them. The average entropy is given as follows,

$$H_{avg} = \frac{1}{K} \sum_{k=1}^{K} H(\mathbf{X}_k). \quad (11)$$

It is worth noting that the uniqueness in terms of μ_{inter} does the same besides assuming independent bits across a multi-bit response. The following example illustrates numerically the difference between the average entropy and the entropy rate of responses to motivate using the entropy rate as the Inter-PUF unpredictability metric instead of the average entropy.

TABLE 2. XORing responses with indices $\mathcal{A} = \{1, 2, 29\}$.

	Challenges	Responses	PUF_1
$\bigoplus$	00001	$\mathbf{X}_1$	01101
$\bigoplus$	00010	$\mathbf{X}_2$	01011
$\bigoplus$	11101	$\mathbf{X}_{29}$	10100
$=$	11110	$\mathbf{X}_{30}$	10010

Example 2. *Suppose there are $K = 2^N$ identically-distributed discrete (N_r-bit) random variables $\mathbf{X}_k$, $1 \leq k \leq K$, and each has an entropy $H(\mathbf{X})$. Assume that there are unknown dependencies between these responses as follows: When XORing some responses, the index of the resulting response $\mathbf{X}_j$ results from XORing the indices of these responses, and vice versa. Table 2 provides a numerical example of this relation. Given a set of indices $\mathcal{A}$, the relation between responses can be described as follows:*

$$j = \bigoplus_{i \in \mathcal{A}} i \iff \mathbf{X}_j = \bigoplus_{i \in \mathcal{A}} \mathbf{X}_i. \tag{12}$$

Thus, there are only N_r independent responses and the normalized joint entropy is $\frac{N_r}{K} H(\mathbf{X})$, while the average entropy is $H(\mathbf{X})$. A numerical example can be given by the CRPs in Table 1, where $N = N_r$. There are only 5 independent responses of a total of 32 responses. For example, $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_4$, $\mathbf{X}_8$, $\mathbf{X}_{16}$ are independent responses. Thus, the normalized joint entropy is $\frac{5}{32} H(\mathbf{X}) = \frac{25}{32}$ bits/response, while the average entropy is $H(\mathbf{X}) = 5$ bits/response. Clearly, the average entropy fails to account for dependencies between different responses.

Thus, in the case of k i.i.d. responses, $kH(\mathbf{X})$ bits suffices. Otherwise, for non-i.i.d. responses, the entropy rate, $H(\mathcal{X})$, suffices. To sum up, besides being a natural, sufficient, and consistent measure, the entropy rate: 1) precisely measures the unpredictability in a PUF class and excludes any dependency, 2) measures how much unpredictability is left in the k^{th} response after observing $(k-1)$ responses, and 3) gives the maximum number of CRPs that can be used before an attacker can successfully guess all the remaining CRPs.

In the case of using the empirical distribution of CRPs, we estimate the entropy rate, as we show in section 4, via the following finite-order quantities based on the available number of samples and the processing cost:

1) The k^{th}-order normalized joint entropy of CRPs.
2) Average of marginal entropy of CRPs. (Assuming CRPs are independent.)
3) The n^{th}-order entropy over bits and averaged over CRPs.
4) Average entropy over bits and CRPs. (Assuming both bits and CRPs are independent.)

Remark 3. *The last quantity measures the uniqueness across chips. However, many works in the literature replace this quantity with the average HD over different responses denoted as μ_{inter}. In subsection 5.1, we show that the second quantity above is the correct measure for uniqueness. Even the third and the last quantities are more precise than the average HD.*

The uniqueness is a widely used metric in the PUF literature to measure the independence between different PUF chips [39]. The uniqueness of the k^{th} response $\mathbf{X}_k$ is calculated empirically by the average pairwise normalized HD of responses from different PUF chips when queried with the k^{th} challenge, and is given as follows [27], [30], [47]:

$$HD(\mathbf{X}_k) = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \frac{1}{N_r} d_H\left(\mathbf{x}_{k,i}, \mathbf{x}_{k,j}\right), \tag{13}$$

where M is the number of PUFs. The average over different responses is denoted as μ_{inter} in the PUF literature.

Last, we could use the entropy rate across PUF chips to measure the number of independent PUF chips, which is sufficient to completely identify a PUF class. In other words, it measures the number of independent PUF chips such that there is no new information left in any other chip.

$$H(\mathcal{PUF}) = \lim_{k \to \infty} \frac{1}{k} H(PUF_1, PUF_2, \ldots, PUF_k), \tag{14}$$

and

$$H(\mathcal{PUF}) = \lim_{k \to \infty} H(PUF_k | PUF_1, PUF_2, \ldots, PUF_{k-1}). \tag{15}$$

In appendix section B, we discuss the analogy between strong PUFs and one-way functions (OWF) such that we could leverage the entropy rate estimation to analyze the unpredictability of OWFs.

4. Finite-Order Entropy Rate Estimation

One main limitation is that the joint distribution of responses is not available for any existing PUF class to the authors' knowledge. This requires us to estimate such performance metrics instead of calculating them directly using probability distributions; future work will focus on deriving the probability distributions to accurately calculate different performance metrics. Nonetheless, estimating these metrics requires a sufficient number of samples that may be obtained either from hardware implementation, simulation, or both, which is not an easy task in general. We do not make any implicit assumptions about independent CRPs or chips but rather we calculate a finite-order entropy rate based on the available sample size and the estimator performance to avoid estimation bias.

In the absence of the joint distribution of responses, we want to estimate the entropy rate, which is an asymptotic notion and is a difficult task in general. Existing work proposes and analyzes entropy and entropy rate estimators for special cases of stochastic processes such as stationary and ergodic processes. Plug-in estimators and data compression techniques can be used for this task. However, the effective alphabet size grows exponentially with the number of random variables of a stochastic process. On the other hand, entropy estimation requires a sample complexity $\mathcal{O}(\frac{|\mathcal{X}|}{\log|\mathcal{X}|})$, while optimal compression techniques need more than $|\mathcal{X}|$ samples [48]. In addition, a PUF class may be neither an ergodic nor a stationary process. More importantly, collecting a sufficient number of samples requires fabricating many chips. That is, it is an under-sampled regime.

In the following, we explain how we use the entropy estimators and the structure of our data set to estimate a finite-order of the entropy rate. One parameter of an estimator is the symbol size s bits. Each s-bit symbol that comprises our sample set is transformed into an integer. Thus, the estimator receives a vector of integer samples as input data and returns a scalar estimate of the Shannon entropy. For each PUF size N_r and time delay τ, the collected data are a three-dimensional array of size (M, L, N_r). Here, M is the number of different PUF chips realized experimentally and $L \leq K = 2^N$ is the number of N_r-bit responses generated from L different challenges. The structure of this data block is

$$\begin{matrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \mathbf{x}_{1,3} & \cdots & \mathbf{x}_{1,M} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \mathbf{x}_{2,3} & \cdots & \mathbf{x}_{2,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{L,1} & \mathbf{x}_{L,2} & \mathbf{x}_{L,3} & \cdots & \mathbf{x}_{L,M} \end{matrix} \tag{16}$$

We denote $\mathbf{x}_{l,m} = \{x^n_{l,m}\}$ as the l^{th} N_r-bit response of the m^{th} PUF chip, where $1 \leq n \leq N_r$ and $1 \leq l \leq L$.

For PUF sizes of interest for cryptography applications, $N > 32$, the number of symbols would be too large for an accurate estimate of the entropy of the full N-bit response; this is compounded as we attempt to estimate the joint entropy. The k-th order joint entropy has an alphabet size $|\mathcal{X}|^k = 2^{kN_r}$.

The preceding part concerns calculating the unconditional entropy of a single response. However, the entropy rate in (10), calculated over observed responses, is the true metric of interest. Therefore, we need a method to estimate the correlations across responses and PUFs, subject to a constraint on s, which is the largest symbol size for which the entropy can be effectively estimated with a negligible estimation bias. Thus, we sample s bits from k responses to form one symbol. This approach has similar limitations as those of calculating the entropy of a single response, extended to correlations between responses. Despite some weaknesses, it is a practical method to move forward and several examples are illustrative. To estimate the entropy of a response using the chunks of 8 bits, we simply set $s = 8$ and $k = 1$. We denote this as an 8-bit first-order entropy rate. Another example is to set s and let $k = 2$. Here, to compose the symbol, $N_b = s/k = s/2$ bits are chosen from each of $k = 2$ responses; with the suitable summation, this yields an estimate of the N_b-bit second-order entropy rate. The N_b bits are randomly chosen among the N_r bits, but in the same order for both responses[2].

Illustrative examples are shown in Figure 2 and Figure 3 where we choose $s = 8$ and $k = 4$ for $N_r = 16$ bits. The

2. We select bits using the same positions in both responses to compose the symbol.

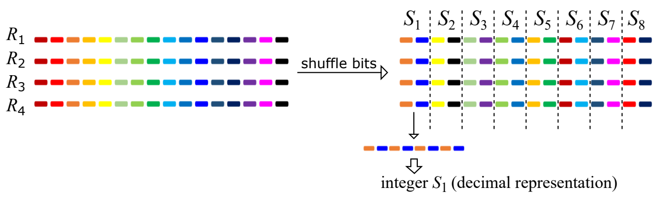
R_1
R_2
R_3
R_4
shuffle bits
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8
integer S_1 (decimal representation)

Figure 2. Creating $N_s = N_r k/s = 8$ symbols of size $s = 8$ bits from a set of $k = 4$ responses of size $N_r = 16$ bits. Left: first four responses of a given PUF chip. The colors represent the position of the bit. The ordering is arbitrary. Right: The bits are shuffled but the bit order is the same across responses. Symbols are created by the concatenation of $N_b = s/k = 2$ bits from each response. The symbols of s bits are transformed into an integer S for entropy estimation.

responses are shown on the left with the colors representing the bit positions: dark red is the first bit and black is the last bit. On the right, the bits are shuffled in a way that the bit order is the same across responses. The symbols are formed by concatenating $N_b = s/k = 2$ bits from each response and then transformed into an integer S (decimal representation). In other words, the first two bits of each shuffled response form the first symbol S_1, and the third and fourth bits of each response form the second symbol S_2, ..., etc. In the entire data set composed of L responses, there are $N_L = L/k$ sets of k responses. For each set, $N_s = N_r k/s$ symbols are formed. For every set of k responses of a given PUF, N_s

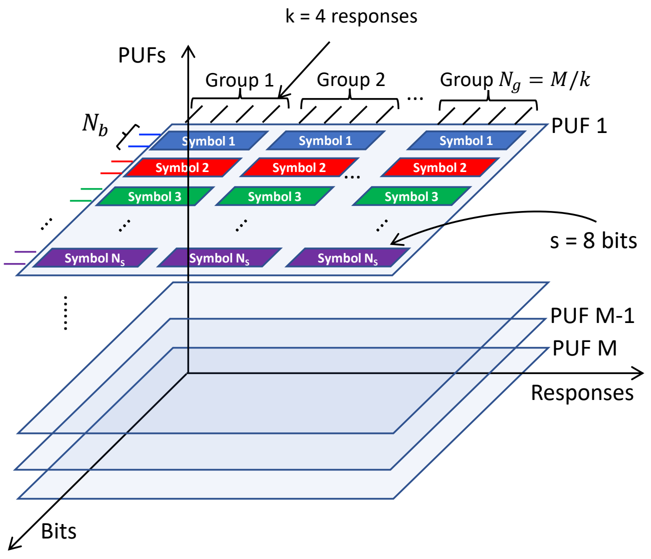

Figure 3. The entropy estimation is done along the vertical axe.

symbols are created and the new block of data that contains integers representing the symbols is formed as:

$$\begin{array}{cccc}
S^{1}_{1,1} & S^{1}_{2,1} & \cdots & S^{1}_{N_s,1} \\
S^{2}_{1,1} & S^{2}_{2,1} & \cdots & S^{2}_{N_s,1} \\
\vdots & \vdots & \ddots & \vdots \\
S^{M}_{1,1} & S^{M}_{2,1} & \cdots & S^{M}_{N_s,1} \\
---- & ---- & ---- & ---- \\
\vdots & \vdots & \ddots & \vdots \\
---- & ---- & ---- & ---- \\
S^{1}_{1,N_L} & S^{1}_{2,N_L} & \cdots & S^{1}_{N_s,N_L} \\
S^{2}_{1,N_L} & S^{2}_{2,N_L} & \cdots & S^{2}_{N_s,N_L} \\
\vdots & \vdots & \ddots & \vdots \\
S^{M}_{1,N_L} & S^{M}_{2,N_L} & \cdots & S^{M}_{N_s,N_L}
\end{array} \tag{17}$$

where $S^{m}_{i,j}$ is the i^{th} symbol of the j^{th} set of k responses for the m^{th} PUF chip. Each column of this data block is treated as a set consisting of M samples over which the entropy is estimated. Thus, applying the entropy estimator results in an $N_L \times N_s$ matrix with entropy estimations given by:

$$\begin{array}{cccc}
\hat{H}_{1,1} & \hat{H}_{2,1} & \cdots & \hat{H}_{N_s,1} \\
\hat{H}_{1,2} & \hat{H}_{2,2} & \cdots & H_{N_s,2} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{H}_{1,N_L} & \hat{H}_{2,N_L} & \cdots & \hat{H}_{N_s,N_L}
\end{array} \tag{18}$$

where $\hat{H}_{i,j}$ is the entropy estimation for the i^{th} symbol of the j^{th} set of k responses calculated over M PUF chips. To obtain the N_b-bit k^{th}-order entropy rate estimate $\hat{H}_{k,N_b}$, 1) divide by number of responses k, 2) sum over N_s symbols, and 3) average over N_L chunks as follows:

$$\hat{H}_{k,N_b} = \frac{1}{kN_L} \sum_{i=1}^{N_s} \sum_{j=1}^{N_L} \hat{H}_{i,j}. \tag{19}$$

5. Entropy for Other PUF Measures

5.1. Uniqueness: Entropy or Hamming Distance?

Entropy is a fundamental quantity in information theory [46] to describe a random variable $\mathbf{X}_k$. It can be defined as a lower bound on the average number of bits required to describe or encode a random variable. Hence, the entropy is **sufficient** to measure the uniqueness of a response. In the following, we give a counterexample that high μ_{inter} does not necessarily imply high entropy, and thus does not imply high independence between chips.

Example 3. *(Entropy is* **precise** *while HD is misleading) We measure the uniqueness of a response in terms of HD and the entropy of the responses in Table 1, e.g., 1 and 2 (the last two rows). In both examples, the response of any PUF chip is selected with equal probability from a set of only two responses,* $\mathbf{X}_{k_1} \in \{10101, 01010\}$ *in e.g.1 and* $\mathbf{X}_{k_2} \in \{11110, 11111\}$ *in e.g.2. This means both have the same underlying distribution and the same structure (only two available symbols, i.e., each half of PUFs in that class cannot be separated). In the first example, the two responses differ in all bit positions. Thus, the normalized HD is either* 0 *or* 1*, which gives the highest* $HD(\mathbf{X}_{k_1}) = 0.5$*. This indicates that the responses are independent. In the second example, only the last bit position differs. Thus, the normalized HD is* 0 *or* $1/5$*, which gives a low* $HD(\mathbf{X}_{k_2}) = 0.1$*. This indicates that responses have a low independence level. On the other hand, guessing such two responses is equivalent to guessing an unbiased flipping coin with equal probability, i.e., one bit only suffices to describe the two different responses. Here, HD gives two different levels of independence between PUF chips,* $HD(\mathbf{X}_{k_1}) = 0.5$ *and* $HD(\mathbf{X}_{k_2}) = 0.1$*, with the same entropy,* $H(\mathbf{X}_{k_1}) = H(\mathbf{X}_{k_2}) = 1$ *bit. In fact, they have the same level of independence. Although in the case of low* μ_{inter} *where designers can identify a weakness in this PUF class, high* μ_{inter} *is not sufficient for high independence (inconclusive) and, thus, is misleading. For a sufficiently large* M*,* μ_{inter} *converges to* 0.5*, indicating independent PUF chips. However, it is unclear whether this implies high entropy or not. Last, the entropy in both cases is* 1 *bit. Here, entropy gives a* **precise** *measure of the number of bits needed to describe uniqueness.*

The previous example shows that high uniqueness requires more bits and hence the entropy is high, and vice versa. That is, the entropy is a **consistent** measure of uniqueness. In addition, unlike entropy, HD considers each bit independently, and hence, it does not capture possible dependencies between response bits in the case of multi-bit strong PUFs. This is typical in many PUF designs, in particular strong PUFs, where each response bit is driven by the same input challenge, and there are possible dependencies across the response bits.

Even for independent response bits[3], HD does not provide any advantage over entropy. The authors in [39] explain the relationship between min-entropy and uniqueness in terms of HD, while the authors in [49] try to estimate the min-entropy of responses using HD and μ_{inter}. In both works, the empirical estimates of the three metrics are directly related. Figure 4 shows that the three metrics are directly related if the response bits are independent. Thus, even for independent bits, there is no need to use the min-entropy or μ_{inter} to estimate the actual entropy.

Figure 5 shows a numerical illustration of the number of samples required for an empirical estimate to converge for the three quantities and that μ_{inter} is a maximum likelihood estimator for both entropy and min-entropy. Miller-Madow entropy estimator [50] converges faster than using μ_{inter}. In section 6, we use different estimators that converge even faster than the Miller estimator. Moreover, μ_{inter} requires a number of computations (computation complexity) that increases quadratically with the number of samples (PUF chips) as $M(M-1)/2$. In other words, μ_{inter} does not provide a better estimate for entropy, uses less number of samples nor conveys any additional measure for uniqueness.

3. Assuming bits are independent is reasonable for memory-based PUFs.

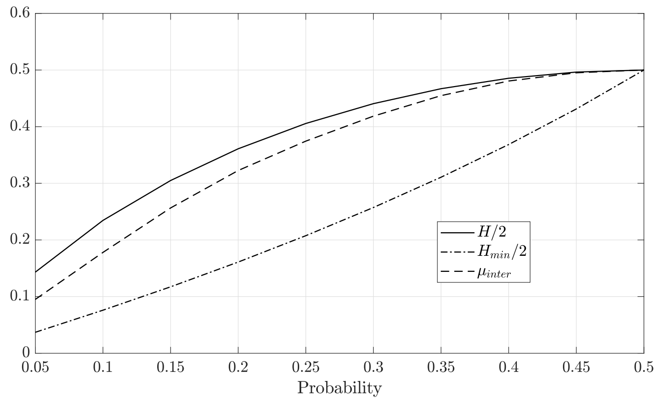

Figure 4. The three metrics are directly related if response bits are independent.

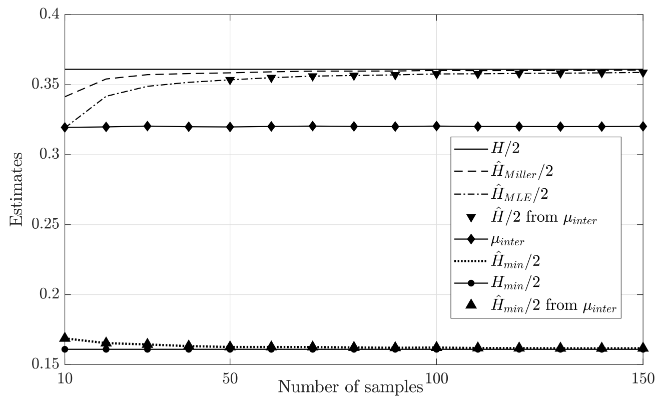

Figure 5. Miller converges faster than μ_{inter} when estimating Entropy and min-entropy.

Example 4. *Another interesting case is when specific PUFs act as counters. Can the entropy be high in this case (as the counter does not repeat), yet the Hamming distance is quite low? Interestingly, both the Hamming distance and entropy is maximum for a counter, $HD = 0.5$ and H=N_r bits, respectively. We note that entropy is used for random sequences that are not deterministic as counters. In subsection 5.2, we provide an idea for testing a single chip against possible deterministic behavior such as acting as counters. Also, we show that a random challenge generator should be used in the enrollment phase rather than a deterministic one, for example, counter in section 6.*

In summary, we justify the use of entropy instead of HD as a uniqueness metric as follows. First, we give a counterexample that high μ_{inter} does not necessarily imply high entropy, and thus does not imply high independence between chips. Second, μ_{inter} does not capture possible dependencies between response bits in the case of multi-bit strong PUFs. Third, the computation complexity of μ_{inter} scales quadratically with the number of samples (PUF chips). Last, even if we assume independent bits in a response, entropy gives a precise measure of independence compared to HD, as shown in Figure 4.

5.2. Testing a PUF chip: Intra-PUF Unpredictability

Now, excluding noise in Definition 2 and focusing on deterministic mapping, a PUF chip could be completely identified by a set of fixed parameters $\mathbf{v}_m$. However, this set of parameters is unknown and attackers try to estimate it[4]. We use the entropy function to indicate whether or not a PUF chip spans a large number of different responses. We exploit the set of responses of a PUF chip to obtain the frequency distribution of the responses and, therefore, the entropy of the PUF chip $H(PUF_m)$ as a measure of the Intra-PUF unpredictability for the m^{th} PUF chip. We show the histogram of the s-order Intra-PUF entropy estimate of HBN-PUF at delay time 10 in Figure 6, which can answer an interesting question: Given a PUF chip, what is the probability that the entropy of this chip deviates from the expected entropy? In other words, what is the percentage of bad PUF chips from a PUF class?

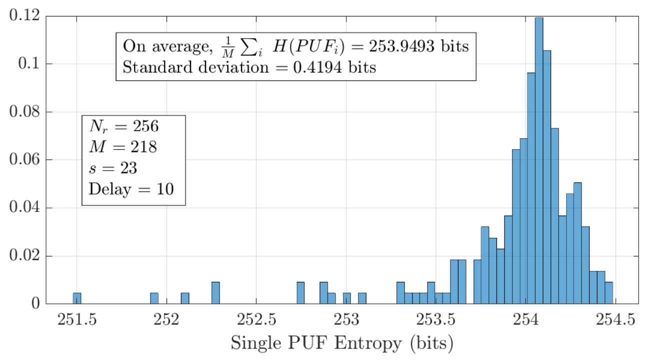

Figure 6. Few HBN PUFs have relatively low entropy.

Remains to 1) detect any unwanted behavior, such as periodic responses or the existence of some deterministic patterns, and 2) prevent a single PUF chip from showing such unwanted behavior. We plot the responses of a bad toy PUF chip in the space of the challenges as in Figure 7 for

4. Hash functions are used to prevent such parameter estimation attacks.

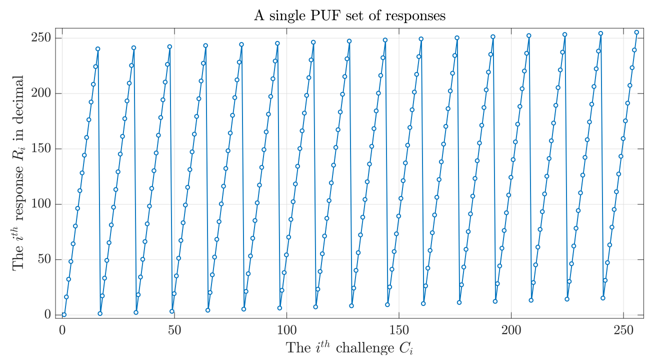

Figure 7. Example of a toy 8-bit PUF that shows a deterministic behavior though its responses (shown in decimal) span all the response space.

example. Fast Fourier transform (FFT) can be used to detect any peak that indicates unwanted deterministic behavior. Then, the number of challenges that can be used should not be more than double the number that results from FFT mimicking the Nyquist sampling theorem. Thus, we still can use a PUF chip that shows an unwanted behavior, but with a limited (reduced) number of CRPs. Future work will further investigate this idea.

6. Experimental Results

In this part, we exploit the theoretical framework to carefully assess the performance of HBN-PUF that benefits from the chaotic behavior of the autonomous Boolean networks and generates responses as fast as just a few nanoseconds. HBN PUF is a promising multi-bit strong PUF class that is developed by our team at the Physics Department, the Ohio State University. The new class, HBN-PUF, benefits from the chaotic behavior of the autonomous (unclocked) Boolean networks and generates responses as fast as just a few nanoseconds. We illustrate the basic idea of the HBN-PUF in subsection A.3.

To measure performance metrics for HBN-PUFs, we focus on collecting many CRPs from different PUF chips. Currently, HBN-PUFs are implemented on field-programmable gate array (FPGA) broads. We implemented 218 Cyclone 10 LP chips (part number 10CL006YU256I7G), collected 65536 randomly chosen challenges/board, and 100 responses/challenge. Thus, we have some practical limitations to collect a sufficient number of CRPs for two reasons: 1) the limited number of FPGA boards, and 2) the large dimensions[5] of HBN-PUFs, which imposes an exponential increase in the number of both required samples and computations. In addition, estimating the joint entropy increases the number of required samples. In summary, we want to tackle the limitations of both the sample size and computation to estimate the unpredictability of the HBN-PUFs as accurately as possible.

Under specific assumptions, we perform the finite-order entropy rate estimation within 1% estimation bias using the available chips. More chips would result in better estimates. Thus, manufacturers can use databases from many chips in the enrollment phase (compared to our limited number 218 chips) to better estimate the unpredictability using our procedures. We use different entropy estimators in the literature such as the maximum likelihood estimator (MLE), the center Dirichlet Mixture (CDM) based estimator [51], the minimax estimators in [52], [53], denoted as JVHW and WY, respectively, and the approximate profile maximum likelihood (PML) [54].

In the following, we use a variety of distributions (e.g., uniform, beta, etc.) to test these estimators. It is worth noting that we do not make any assumption that the distribution of PUF responses is uniform. We use a variety of distributions to test different estimators and choose the one that performs well in general. We show only (as an example) the performance in the case of the uniform distribution (i.e., highest entropy) in Figure 8. As shown in Figure 8, the CDM estimator converges faster specifically for a large support size. For example, the CDM estimator is within a one-bit error compared to the JVHW estimator, which is within a four-bit error when the sample size is $\sqrt{|\mathcal{X}|}$.

5. For example, one PUF has $N = 256$ nodes and generates $N_r = 256$ bits/response, i.e., $256-$bit PUF.

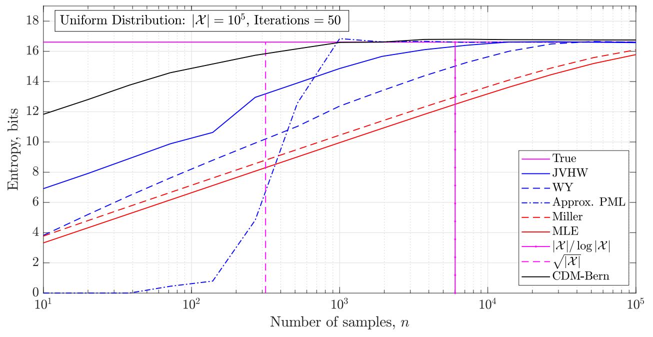

Figure 8. CDM estimator outperforms other estimators for a variety of distributions and large support sizes.

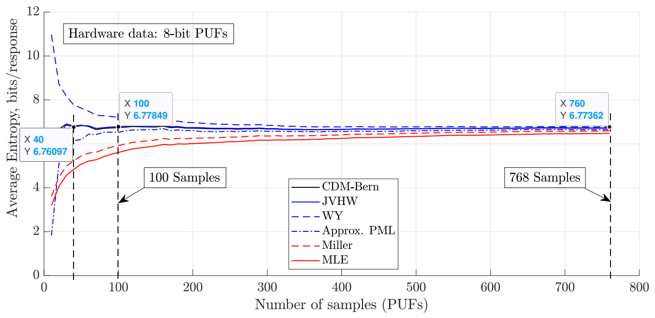

Figure 9. CDM and JVHW estimators converge with a relatively small number of samples compared to other estimators.

In Figure 9, we show the convergence of different estimators for 8-bit HBN-PUFs. Using only 40 PUFs, the CDM and JVHW estimators converge to the true entropy. We use such convergence plots to decide the sufficient number of samples M and the number of bits per symbol s without introducing an estimation bias.

In Figure 10, we plot the convergence of CDM and JVHW estimators for different sizes of uniform bits per symbol. Approximately 200, 6000, and 5×10^8 samples (i.e., PUFs) are needed for sizes 8, 16, and 32, respectively, when using the JVHW estimator compared to 200, 2000, and 2×10^5 samples when using the CDM estimator to estimate the entropy within 1-bit estimation bias. This figure allows us to choose the number of samples needed to estimate the entropy for a specific bias. We note that the JVHW estimator has good performance for small alphabet sizes (e.g., $|\mathcal{X}| = 256$) compared to the CDM estimator.

Now, it is clear that the main problem is overcoming the estimation bias due to the limited number of samples. We use the available samples ($M = 218$) and estimate an upper bound on the entropy. In particular, we assume

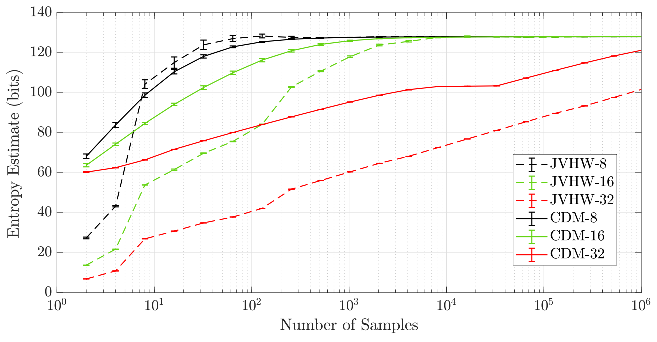

Figure 10. Determining the number of samples needed to estimate the entropy using CDM and JVHW estimators.

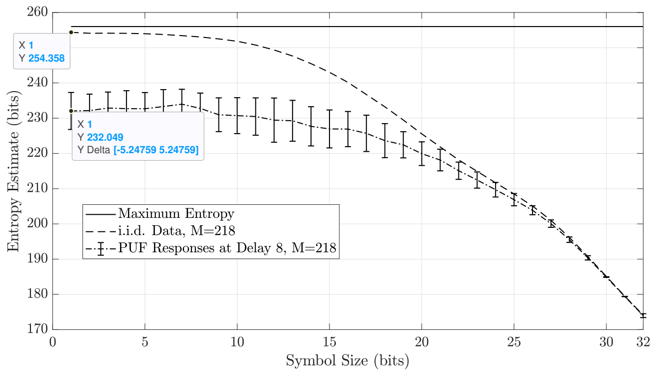

Figure 11. There is no sharp reduction. Thus, $s > 8$ bits are not fully dependent.

that the response is independent on a“symbol-wise” basis. This means we first choose the symbol size s, measured in bits. Within each response, we assume that symbols are independent.

For instance, if we assume a 256-bit PUF, we can divide each response into 32 symbols, each with 8 bits. This is based on the effective upper limit of the alphabet size for which the estimator yields reasonable results. We assume that within each symbol, there may be correlations of up to s order that are captured by the entropy estimation routine. However, we assume that no dependencies exist between symbols, so we can sum the entropy over the 32 symbols of our 256-bit PUF.

This assumption raises two issues. Firstly, the correlation between any two bits may be missed if they are partitioned between two different symbols. We overcome this by Monte Carlo sampling, which shuffles the bits to create different symbols. Secondly, there might be a high correlation between larger than s bits, but the entropy drops to only around s bits. For example, suppose that s is 8 and we estimate the entropy as 250 bits by summing the entropy of 32 symbols, but any 10 bits or larger are fully dependent, such that the actual entropy should drop to around 9 bits. This is not captured by the upper bound.

To solve this issue, we compare the decrease of the upper bound of the entropy when increasing the estimation symbol size s for a fixed sample size and i.i.d. generated bits vs. the PUF responses. A comparable decrease should be due to the small sample estimation bias; otherwise, there are higher-order correlations. Using randomly generated responses for a fixed number of samples ($M = 218$), Figure 11 compares the bias of the entropy estimation as the symbol size increases for both i.i.d. uniform bits and the PUF responses. We use the estimation at $s = 1$ since 1) the estimate is reliable and 2) for larger symbol sizes, we cannot differentiate between the estimation bias and the reduction due to correlations between symbols. There is no sharp reduction, indicating that $s > 8$ bits are not fully dependent.

We normalize the result to get an upper bound as follows:

$$\hat{H}^{norm}_{1,1} = \frac{N_r \times \hat{H}_{1,1}}{\hat{H}_{iid}} = \frac{256 \times 232.049}{254.358} = 233.547 \text{ bits}, \tag{20}$$

where $\hat{H}_{iid}$ is the entropy estimate from i.i.d. data at the same sample and symbol sizes. At 1% estimation bias, i.e., reduction due to sample size, we can use the symbol size up to 10 bits. Thus, to capture the correlation across responses, we calculate the 1-bit tenth-order entropy rate estimate using ten responses to create symbols, i.e., $k = 10$ and $s = 10$. At delay 8, $\hat{H}^{norm}_{10,1} = 231.6895$ bits/response. We use both randomly generated responses and gray-coded responses. The former is more realistic and valid for the astronomical response space. The latter captures the true entropy of the PUF.

In Figure 12, we plot the best-case normalized entropy rate estimate $\hat{H}^{norm}_{10,1}/N_r$ for different delays to decide the optimal delay time. In the best case, we assume that there is no further reduction due to using more than ten responses. Since the estimated entropy rate is for noisy responses, we also plot the entropy of μ^{maj}_{intra}, which is the reduction due to error correction where μ^{maj}_{intra} is the average error per bit but after the majority voting of responses. Here, we assume that two forms of error correction can be performed. The first is majority voting after querying the PUF many times for the same challenge. This is done in both the enrollment and the utilization phases. The second is a suitable error-correcting code where helper data are used.

In the worst case where we assume that there is always a further linear reduction with more responses, we may use only $\hat{H}^{norm}_{1,1} \times \frac{k}{\hat{H}^{norm}_{1,1} - \hat{H}^{norm}_{10,1}}$ noisy responses, where $\hat{H}^{norm}_{1,1} - \hat{H}^{norm}_{10,1}$ is the reduction due to the use of ten responses. For example, at delay 8, we may use only noisy responses $233.547 \times \frac{10}{1.8575} = 1257$ in the worst-case scenario. In the best case, an attacker might be able to achieve a cloning error down to the probability of error for the binary entropy of $\hat{H}^{norm}_{k,N_b}/N_r$ per bit using only k CRPs from the attacked PUF and other $M-1$ PUFs. For example, at delay 8, the cloning error could be reduced to 32%, the probability of error for the binary entropy of $231.6895/256 = 0.905$

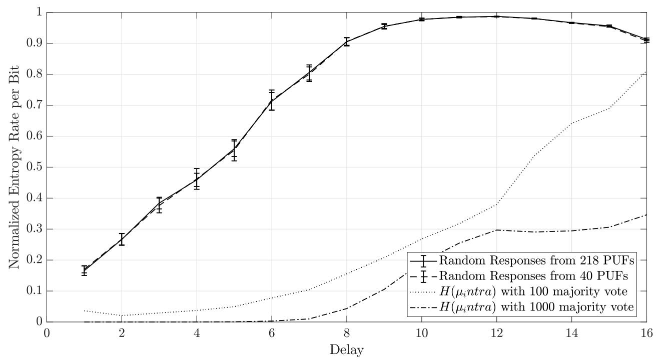

Figure 12. Normalized entropy rate peak is at delays $10 - 12$, similar to using 40 PUFs.

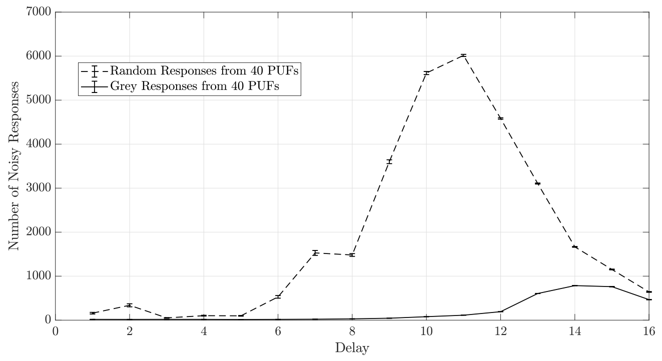

Figure 13. Worst-case estimate is 6000 noisy responses at delay 11 for random responses.

per bit, using only ten randomly generated CRPs from the attacked PUF and another 217 PUFs. Thus, a PUF of entropy 0.905 per bit is due to a bias of 0.32 such that the attacker generates a response of all ones to achieve a probability of cloning error 0.32. Our approach gives a precise measure of unpredictability and does not only indicate whether a PUF is susceptible to a specific attack or not.

In Figure 14, we plot the best-case estimate $\frac{\hat{H}^{norm}_{k,N_b}}{N_r} - H(\mu^{maj}_{intra})$ that corresponds to the remaining entropy per bit after using public helper data to correct errors. This precisely determines the optimal querying time. The entropy rate of approximately $\frac{231.6895}{256} - 0.043 = 0.862$ per bit can be achieved using the 1000 majority votes. The corresponding cloning error is shown in Figure 15. For randomly chosen responses, the cloning error is around 29%, while using gray-coded responses reduces the cloning error to 18%.

We conclude the following lessons from the previous part. A random challenge generator should be used in the enrollment phase rather than a deterministic one, e.g., counter. Since correlated challenges (that have a small Hamming distance) could generate correlated responses, we suggest using a threshold on the minimum HD, for example, 40 bits, between randomly generated challenges to reduce the PUF predictability. Furthermore, if the attacker has access to the PUF, a good approach is to use gray-coded challenges to query the PUF and get as many responses as possible.

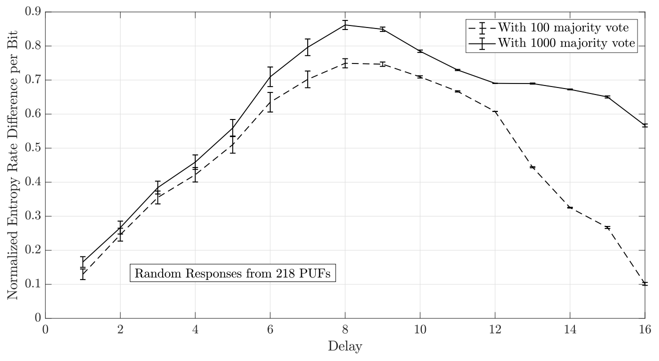

Figure 14. Normalized entropy rate peak is moved back to delay 8.

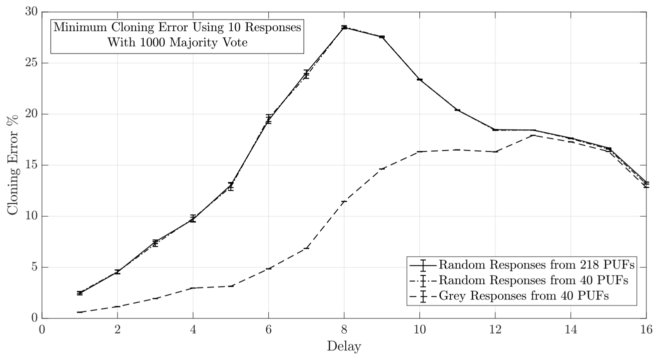

Figure 15. Minimum cloning error at delay 8 is around 29%.

Then, the attacker may be able to clone the PUF using his gray-coded CRPs database with a low probability of error.

7. Conclusion

We explain the parameters and sources of randomness that affect the unpredictability and reliability of strong PUFs. Our proposed attack model uses chips jointly to predict responses with a minimized cloning error. We show how to measure and estimate the unpredictability of a multi-bit strong PUF class via the entropy rate. Moreover, entropy outperforms the Hamming distance when measuring the uniqueness of PUFs, even with the same number of samples. Efficient min-max entropy estimators are used to estimate a finite-order entropy rate of the Hybrid Boolean Network PUF. Experimental results show its high unpredictability and resistance to ML attacks. Proposing an entropy estimator for the undersampled regime is crucial to the PUF community due to the scarcity of available data to analyze. Additionally, considering Bayesian estimators that are relevant to the nature of binary responses can improve the existing estimators.

References

[1] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, “Identification and Authentication of Integrated Circuits,” *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[2] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *2007 44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[3] A. Bendary, C. E. Koksal, D. Canaday, and A. Pomerance, "Unconditional Authentication for Constrained Applications via Strong PUFs," in *2021 IEEE Conference on Communications and Network Security (CNS)*, 2021, pp. 272–280.

[4] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 33–44.

[5] 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 - CHES 2007*, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–80.

[6] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, "A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1137–1150, 2015.

[7] N. Charlot, D. Canaday, A. Pomerance, and D. J. Gauthier, "Hybrid Boolean Networks as Physically Unclonable Functions," *IEEE Access*, vol. 9, pp. 44 855–44 867, 2021.

[8] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[9] V. Van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware Intrinsic Security from D Flip-Flops," in *Proceedings of the Fifth ACM Workshop on Scalable Trusted Computing*, ser. STC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 53–62.

[10] U. Rührmair, J. Sölter, and F. Sehnke, "On the Foundations of Physical Unclonable Functions," *IACR Cryptol. ePrint Arch.*, vol. 2009, p. 277, 2009.

[11] U. Rührmair and M. van Dijk, "PUFs in Security Protocols: Attack Models and Security Evaluations," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 286–300.

[12] S. Choi, D. Kim, Y. Choi, W. Sun, and H. Shin, "Multibit-Generating Pulsewidth-Based Memristive-PUF Structure and Circuit Implementation," *Electronics*, vol. 9, no. 9, 2020.

[13] C. Xu, J. Zhang, M.-K. Law, X. Zhao, P.-I. Mak, and R. P. Martins, "An N X N Multiplier-Based Multi-Bit Strong PUF using Path Delay Extraction," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[14] V. K. Rai, S. Tripathy, and J. Mathew, "Design and Analysis of Reconfigurable Cryptographic Primitives: TRNG and PUF," *J Hardw Syst Secur*, vol. 5, p. 247–259, 2021.

[15] T. Kroeger, W. Cheng, S. Guilley, J.-L. Danger, and N. Karimi, "Enhancing the Resiliency of Multi-bit Parallel Arbiter-PUF and Its Derivatives Against Power Attacks," in *Constructive Side-Channel Analysis and Secure Design*, S. Bhasin and F. De Santis, Eds. Cham: Springer International Publishing, 2021, pp. 303–321.

[16] I. Baturone, R. Roman, and A. Corbacho, "A Unified Multibit PUF and TRNG Based on Ring Oscillators for Secure IoT Devices," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6182–6192, 2023.

[17] C.-C. Lin and M.-S. Chen, "Learning from Output Transitions: A Chosen Challenge Strategy for ML Attacks on PUFs," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2023, pp. 1–6.

[18] N. Wisiol, C. Mühl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Rührmair, "Splitting the Interpose PUF: A Novel Modeling Attack Strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, p. 97–120, Jun. 2020.

[19] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine Learning Resistant Strong PUF: Possible or a Pipe Dream?" in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 19–24.

[20] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 237–249.

[21] F. Ganji, S. Tajik, P. Stauss, J.-P. Seifert, M. M. Tehranipoor, and D. Forte, "Rock'N'Roll PUFs: Prafting Provably Secure PUFs from Less Secure Ones (Extended Version)," *Journal of Cryptographic Engineering*, vol. 11, pp. 105 – 118, 2020.

[22] S. Kumar and M. Niamat, "Machine Learning based Modeling Attacks on a Configurable PUF," in *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, 2018, pp. 169–173.

[23] F. Ganji, S. Tajik, F. Fäßler, and J.-P. Seifert, "Strong Machine Learning Attack Against PUFs with No Mathematical Model," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 391–411.

[24] D. Canaday, W. A. S. Barbosa, and A. Pomerance, "A novel attack on machine-learning resistant physical unclonable functions," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 25–28.

[25] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing Techniques for Hardware Security," in *2008 IEEE International Test Conference*, Oct 2008, pp. 1–10.

[26] P. Tuyls, B. Škorić, S. Stallinga, A. H. M. Akkermans, and W. Ophey, "Information-Theoretic Security Analysis of Physical Uncloneable Functions," in *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, ser. FC'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 141–155.

[27] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs," in *2010 International Conference on Reconfigurable Computing and FPGAs*, 2010, pp. 298–303.

[28] F. Armknecht, R. Maes, A. Sadeghi, F. Standaert, and C. Wachsmann, "A Formalization of the Security Features of Physical Functions," in *2011 IEEE Symposium on Security and Privacy*, May 2011, pp. 397–412.

[29] R. Nithyanand and J. Solis, "A Theoretical Analysis: Physical Unclonable Functions and the Software Protection Problem," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 1–11.

[30] A. Maiti, V. Gunreddy, and P. Schaumont, *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*. New York, NY: Springer New York, 2013, pp. 245–267.

[31] A. Schaub, O. Rioul, and J. J. Boutros, "Entropy Estimation of Physically Unclonable Functions via Chow Parameters," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE Press, 2019, p. 698–704.

[32] 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, pp. 121–137, 2016.

[33] F. Ganji, D. Forte, and J.-P. Seifert, "PUFmeter a Property Testing Tool for Assessing the Robustness of Physically Unclonable Functions to Machine Learning Attacks," *IEEE Access*, vol. 7, pp. 122 513–122 521, 2019.

[34] S. Zhao, J. Lin, W. Li, and B. Qi, "Research on Root of Trust for Embedded Devices based on On-Chip Memory," in *2021 International Conference on Computer Engineering and Application (ICCEA)*, 2021, pp. 501–505.

[35] N. Karimian and F. Tehranipoor, "How to Generate Robust Keys from Noisy DRAMs?" in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 465–469.

[36] J. Francq and G. Parlier, *Secure Key Generator Using a Loop-PUF*. Cham: Springer International Publishing, 2015, pp. 143–173.

[37] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling Poses Strict Bounds on Usability," in *2012 IEEE international workshop on Information forensics and security (WIFS)*. IEEE, 2012, pp. 37–42.

[38] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 283–301.

[39] C. Gu, W. Liu, N. Hanley, R. Hesselbarth, and M. O'Neill, "A Theoretical Model to Link Uniqueness and Min-Entropy for PUF Evaluations," *IEEE Transactions on Computers*, vol. 68, no. 2, pp. 287–293, Feb 2019.

[40] J. Delvaux, D. Gu, and I. Verbauwhede, "Upper Bounds on the Min-Entropy of RO Sum, Arbiter, Feed-Forward Arbiter, and S-ArbRO PUFs," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, 2016, pp. 1–6.

[41] M. Shiozaki, Y. Hori, and T. Fujino, "Entropy Estimation of Physically Unclonable Functions with Offset Error," Cryptology ePrint Archive, Paper 2020/1284, 2020.

[42] Q. Wang and G. Qu, "A Silicon PUF Based Entropy Pump," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 402–414, 2019.

[43] W. Che, V. K. Kajuluri, M. Martin, F. Saqib, and J. Plusquellic, "Analysis of Entropy in a Hardware-Embedded Delay PUF," *Cryptography*, vol. 1, no. 1, 2017.

[44] M. S. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle, "NIST, SP 800–90B: Recommendation for the Entropy Sources Used for Random Bit Generation," Tech. Rep, National Institute for Standards and Technology, Tech. Rep., 2018.

[45] J. Wang, Z. Chen, and J. Lu, "A Fault Propagation Model for FPGA Netlist with Un-Reconvergence Paths," in *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, 2018, pp. 1015–1019.

[46] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, Inc., 2012.

[47] L. Feiten, M. Sauer, and B. Becker, "On Metrics to Quantify the Inter-Device Uniqueness of PUFs," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 320, 2016.

[48] S. Verdú, "Empirical Estimation of Information Measures: A Literature Guide," *Entropy*, vol. 21, no. 720, 2019.

[49] H. Liu, W. Liu, Z. Lu, Q. Tong, and Z. Liu, "Methods for Estimating the Convergence of Inter-Chip Min-Entropy of SRAM PUFs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 593–605, 2018.

[50] G. Miller, "Note on the Bias of Information Estimates," *Information theory in psychology: Problems and methods*, 1955.

[51] E. W. Archer, I. M. Park, and J. W. Pillow, "Bayesian Entropy Estimation for Binary Spike Train Data Using Parametric Prior Knowledge," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1700–1708.

[52] J. Jiao, K. Venkat, Y. Han, and T. Weissman, "Minimax Estimation of Functionals of Discrete Distributions," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2835–2885, 2015.

[53] Y. Wu and P. Yang, "Minimax Rates of Entropy Estimation on Large Alphabets via Best Polynomial Approximation," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3702–3720, 2016.

[54] D. S. Pavlichin, J. Jiao, and T. Weissman, "Approximate Profile Maximum Likelihood," *J. Mach. Learn. Res.*, vol. 20, pp. 122:1–122:55, 2019.

[55] M. Khalafalla and C. Gebotys, "PUFs Deep Attacks: Enhanced Modeling Attacks Using Deep Learning Techniques to Break the Security of Double Arbiter PUFs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 204–209.

[56] A. Maiti and P. Schaumont, "Improved Ring Oscillator PUF: An FPGA-Friendly Secure Primitive," *J Cryptol*, vol. 24, p. 375–397, 2011.

[57] R. D.P., "(2015) Summary and Outlook. In: Dynamics of Complex Autonomous Boolean Networks," in *Springer Theses (Recognizing Outstanding Ph.D. Research). Springer, Cham*, Jan. 2015.

[58] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[59] L. A. Levin, "The Tale of One-Way Functions," *Problems of Information Transmission*, vol. 39, no. 1, pp. 92–103, 2003.

[60] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

Appendix A. State-of-the-Art Strong PUFs

In the following, we illustrate the definition of the PUF class and the different parameters that affect a PUF class by providing examples from state-of-the-art strong PUFs.

A.1. Arbiter PUF

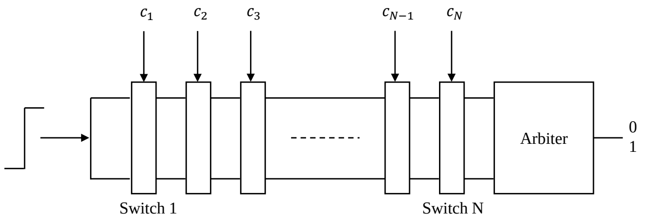

Figure 16. The arbiter PUF circuit [1].

The arbiter PUF in Figure 16 is the first silicon-based PUF. It consists of N switches, each has two paths with delays $\tau^m_{n,1}$ and $\tau^m_{n,2}$, respectively, $1 \leq n \leq N$. The challenge is applied to the switches and creates two different paths with delays:

$$\tau^m_1 = \sum_{n=1}^{N} c_{k,n} \times \tau^m_{n,1} + \sum_{n=1}^{N} \overline{c}_{k,n} \times \tau^m_{n,2}, \tag{21}$$

and

$$\tau^m_2 = \sum_{n=1}^{N} \overline{c}_{k,n} \times \tau^m_{n,1} + \sum_{n=1}^{N} c_{k,n} \times \tau^m_{n,2}. \tag{22}$$

where $\overline{c}_{k,n}$ is the bit complement of $c_{k,n}$.

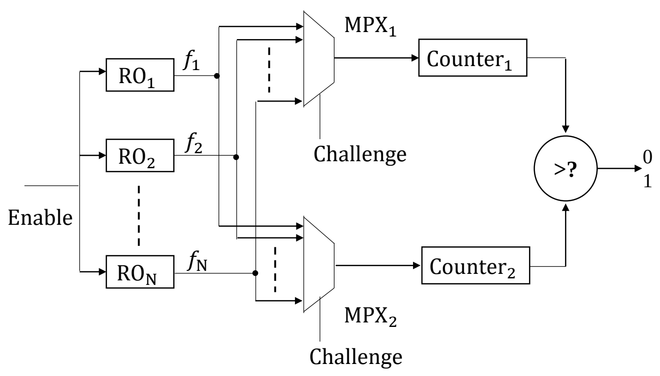

Figure 17. The ring oscillator PUF circuit [56].

An input signal $x(t)$ is applied to both paths and an arbiter (a latch) is placed at the end of the two paths and decides which signal, $x(t-\tau_1^m)$ or $x(t-\tau_2^m)$, reaches it first as follows:

$$f\left(\mathbf{c}_k, \mathbf{v}_m, \boldsymbol{W}(t)\right) = \begin{cases} 1, & y(t-\tau_1^m) \geq y(t-\tau_2^m), \\ 0, & \text{otherwise}, \end{cases} \tag{23}$$

where $y(t-\tau_i^m) = x(t-\tau_i^m) + w_i(t)$, $i \in \{1,2\}$, $\mathbf{v}_m = \left[\tau_{1,1}^m \;\ldots\; \tau_{i,j}^m \;\ldots\; \tau_{N,2}^m\right] \in \mathcal{V} \subset \mathbb{R}^{2N}$, i.e., $N_v = 2N$, is a vector of paths' delays that represents a realization of the random variations due to the manufacturing process, which affects the response of the m^{th} chip, and $\tau_{i,1}^m \sim$ log-normal$(\mu_\tau, \sigma_\tau^2)$. Thermal noise is represented by $\boldsymbol{W}(t) = [w_i(t)] \in \mathcal{W} \subset \mathbb{R}^2$, that is, $N_w = 2$.

Remark 4. *The thermal noise, $w_1(t)$ and $w_2(t)$, changes the actual path delay, and therefore, $y(t-\tau_i^m)$ in* (23) *can be rewritten equivalently as $x(t-\tau_i^m-\tau_i^w(t))$, where $\tau_i^w(t)$ is the corresponding change in path delay due to thermal noise.*

The output of the arbiter PUF is only one bit per challenge, i.e., $N_r = 1$. Therefore, a maximum of 2^N bits can be extracted from the arbiter PUF. Besides, the simple mapping of the arbiter PUF, as in (21)-(23), makes it vulnerable to modeling attacks (e.g., [18], [55]). To enhance the unpredictability of the arbiter PUF, multiple PUFs can be XORed together as follows:

$$X_{k,m'} = \bigoplus_{m=1}^{M} X_{k,m}, \tag{24}$$

where $X_{k,m'}$ is the response of the m' chip and the k^{th} challenge results from XORing M PUFs on the chip.

A.2. Ring Oscillator PUF

The ring oscillator (RO) Figure 17 PUF consists of N ring oscillators, each oscillates at a different frequency due to the variations in the manufacturing process. The challenge is applied to two multiplexers to choose a pair of oscillators and compare their frequencies via a counter. Equivalently, the mapping can be expressed in terms of the total delay of each oscillator [56]. Thus, the RO PUF mapping can be given as follows:

$$f\left(\mathbf{c}_k, \mathbf{v}_m, \boldsymbol{W}(t)\right) = \begin{cases} 1, & \tau_j^m + \tau_j^w(t) \leq \tau_{j'}^m + \tau_{j'}^w(t), \\ 0, & \text{otherwise}, \end{cases} \tag{25}$$

where

$$\tau_j^m = \sum_{n=1}^{N_g} \tau_{n,j}^m, \tag{26}$$

the pair of oscillators j and j' are chosen by the challenge $\mathbf{c}_k$, N_g is the number of gates in each oscillator, $\mathbf{v}_m = \left[\tau_{1,1}^m \;\ldots\; \tau_{n,j}^m \;\ldots\; \tau_{N_g,N}^m\right] \in \mathcal{V} \subset \mathbb{R}^{NN_g}$, i.e., $N_v = NN_g$, is a vector of paths' delays that represents a realization of the random variations due to the manufacturing process, which affects the response of the m^{th} chip, and $\tau_{n,j}^m \sim$ log-normal$(\mu_\tau, \sigma_\tau^2)$. The thermal noise is represented by $\boldsymbol{W}(t) = [w_j(t)] \in \mathcal{W} \subset \mathbb{R}^2$, i.e., $N_w = 2$, and $\tau_j^w(t)$ is the corresponding change in the path delay due to the thermal noise $w_j(t)$. Similar to the Arbiter PUF, the output of the RO PUF is only one bit per challenge, i.e., $N_r = 1$.

A.3. Hybrid Boolean Network PUF

The HBN-PUF consists of N nodes and the output of each node is fed back autonomously, i.e., without a clock, with the output from two randomly selected nodes. The initial states of the nodes are set by the input challenge, and the transient response is captured using the global clock. The HBN-PUF mapping can be given as follows:

$$f\left(\mathbf{c}_k(t), \mathbf{v}_{m,n}, \boldsymbol{W}_n(t)\right) = \begin{cases} 1, & x_{k,m}^n(t) \geq 0.5, \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

where $x_{k,m}^n(t)$ is the continuous-time version of the n^{th} bit response of the m^{th} chip and can be represented using differential equations as follows [57]:

$$\tau_{LP}^{k,n} \frac{d}{dt} x_{k,m}^n(t) = -c_{k,n}(t) + a_{k,n}^m(t) + w_n(t), \tag{28}$$

where

$$a_{k,i}^m(t) = g\left(c_{k,i'}^w(t-\tau_{i'}^{m,i}), c_{k,i}^w(t-\tau_i^{m,i}), c_{k,i''}^w(t-\tau_{i''}^{m,i})\right), \tag{29}$$

$$c_{k,i}^w(t-\tau_i^{m,n}) = c_{k,i}(t-\tau_i^{m,n}) + w_i^n(t),\ i \in \{i', n, i''\}, \tag{30}$$

and $g(\cdot)$ is the continuous-time version of the Boolean XOR function, i', i'' are randomly selected nodes, the low-pass delay is related to the rise time of the n^{th} bit of the k^{th} challenge as $\tau_{LP}^{k,n} = \tau_r^{k,n}/\ln 2$, $\mathbf{v}_{m,n} = [\tau_i^{m,n}] \subset \mathbb{R}^{3\times 1}$, is a vector of paths' delays that represents a realization of the random variations due to the manufacturing process, which affects the n^{th} response bit of the m^{th} chip, and $\tau_i^{m,n} \sim$ log-normal$(\mu_\tau, \sigma_\tau^2)$. The vector $\mathbf{v}_m = [\mathbf{v}_{m,n}] \subset \mathcal{V}$ is a vector of all delays on the m^{th} chip, and here $N_v = 3N$. The thermal noise that affects the n^{th} bit response is represented

TABLE 3. COMPARING THE DIMENSIONS OF THE RANDOM PARAMETERS THAT AFFECT EACH CLASS.

	Arbiter	RO	HBN
N_v	$2N$	NN_g	$3N$
E_v	$2N$	$2N_g$	3
N_w	2	2	$4NN_{st}$
E_w	2	2	$4N_{st}$
N_r	1	1	N

by $\boldsymbol{W}_n(t) = [w_1^n(t)\ \dots\ w_3^n(t)\ w_n(t)] \subset \mathbb{R}^{1\times 4}$. The vector $\mathbf{W}(t) = [\mathbf{W}_n(t)] \subset \mathcal{W}$ has a dimension of $4N$, i.e., $N_w = 4N$. The output of the HBN-PUF is N bits per challenge, i.e., $N_r = N$ and a maximum of $N2^N$ bits can be extracted from the HBN-PUF.

Note that the power level of the noise is negligible compared to the digital signal level. For example, the noise power spectral density is -174 dBm/Hz at room temperature 290°K. However, when the sampling time is within the rise/fall time of the digital signal, the output is sensitive to thermal noise. This is the case for ring oscillators and autonomous Boolean networks in general.

The feedback in each node can be decomposed and viewed as cascaded N_{st} stages. In particular, the input to the n^{th} node in the i^{th} stage is fed from the output of the nodes i', n, i'' of stage $(i-1)$. The number of stages is given by the measurement time divided by the circuit delay (0.5 ns as in [7]). For cascaded identical and independent binary symmetric channels (BSC), each with a probability of error p, the equivalent single BSC has an error probability $\frac{1}{2}(1-(1-2p)^i)$ and $\lim_{i\to\infty}\frac{1}{2}(1-(1-2p)^i) = \frac{1}{2},\ p \neq 0, 1$. This may explain why the error dominates after a long measurement time. Furthermore, the error $\frac{1}{2}(1-(1-2p)^{N_{st}})$ is equivalent to integrating the noise in the differential equation in (28). Thus, the dimensions of $\mathbf{W}(t)$ can be modified to be $N_w = 4NN_{st}$.

The unpredictability of a PUF class depends on the size of the random variations, in terms of N_v and the structure/mapping of the class. The larger the size of the variations and the more complex the structure of the PUF class, the higher the unpredictability of the PUF class. Similarly, the reliability of the PUF class is affected by the thermal noise in terms of N_w. Although introducing more elements in the PUF structure may increase the unpredictability, it may also increase the delay and the cost, and more importantly decrease the reliability. In Table 3, the effective dimensions of the random parameter vector and the noise vector that affect one bit of the output response are denoted as E_v and E_w, respectively.

Appendix B. On the Relation to One-Way Functions

In the early literature, PUFs were referred to as physical OWFs. A OWF is a function that is "easy" to compute in the forward direction, but very "hard" to invert, even on average.

TABLE 4. ONE-WAY FUNCTION VS. PHYSICALLY UNCLONABLE FUNCTION.

	Well-defined mapping	Random parameters	Input	Output
OWF	Yes	Public	Secret	Public
PUF	Not necessarily	Unknown	Public	Secret

Definition 5 (One-way function [58])**.** *A function $f : \{0,1\}^N \to \{0,1\}^N$ is a one-way function (OWF) if:*

1) *f is computable in time $poly(N)$.*
2) *For some $s(N) = N^{\omega(1)}$ and $\epsilon(N) = \frac{1}{N^{\omega(1)}}$, and all nonuniform algorithms A running in time $s(N)$, we have*

$$Pr\left[A\left(f(\mathbf{X})\right) \in f^{-1}\left(f(\mathbf{X})\right)\right] \leq \epsilon(N), \tag{31}$$

where the probability is taken over $\mathbf{X}$ and the randomness of A.

A candidate for OWFs is universal hashing, though OWFs have not been proven to exist yet [59]. An example of a universal class of hash functions is given as follows [60]:

$$\mathcal{H}_{p,m} = \left\{h_{a,b} : a \in \{1,\dots,p-1\} \text{ and } b \in \{0,\dots,p-1\}\right\}, \tag{32}$$

where

$$h_{a,b}(k) = ((ak+b) \mod p) \mod m, \tag{33}$$

i.e., $h_{a,b}$ maps a given key $k \in \{0,\dots,p-1\}$ into the range $\{0,\dots,m-1\}$, and $p > m$ is a prime number.

Although PUFs are preferred to be "hard" to invert on average, as in (31), there are fundamental differences between PUFs and OWFs. While the input to a OWF is secret, the mapping (which is well defined), the used random parameters, and the output are public. However, an attacker should not be able to obtain the secret input in a polynomial time of n. Unlike OWFs, only the challenge to a PUF is sent in public, while the mapping is not necessarily well defined, the random parameters are unknown, and the response is secret in many use cases. Thus, an attacker has to accurately model the PUF class mapping (e.g., with the aid of a sufficient number of implemented PUF chips) and estimate the random parameters of each PUF chip (using previously seen CRPs from that chip). The main differences between OWFs and PUFs are summarized in Table 4.