Security Analysis of Arbiter PUF and Its Lightweight Compositions Under Predictability Test* [Revised Version]

PHUONG HA NGUYEN, DURGA PRASAD SAHOO, RAJAT SUBHRA CHAKRABORTY, and DEBDEEP MUKHOPADHYAY†, Indian Institute of Technology Kharagpur

Unpredictability is an important security property of Physically Unclonable Function (PUF) in the context of statistical attacks, where the correlation between challenge-response pairs is explicitly exploited. In existing literature on PUFs, Hamming Distance test, denoted by HDT(t), was proposed to evaluate the unpredictability of PUFs, which is a simplified case of the Propagation Criterion test PC(t). The objective of these testing schemes is to estimate the output transition probability when there are t or less than t bits flips, and ideally, this probability value should be 0.5. In this work, we show that aforementioned two testing schemes are not enough to ensure the unpredictability of a PUF design. We propose a new test which is denoted as HDT($\mathbf{e}$, t). This testing scheme is a fine-tuned version of the previous schemes, as it considers the flipping bit pattern vector $\mathbf{e}$ along with parameter t. As a contribution, we provide a comprehensive discussion and analytic interpretation of HDT(t), PC(t) and HDT($\mathbf{e}$, t) test schemes for Arbiter PUF (APUF), XOR PUF and Lightweight Secure PUF (LSPUF). Our analysis establishes that HDT($\mathbf{e}$, t) test is more general in comparison with HDT(t) and PC(t) tests. In addition, we demonstrate a few scenarios where the adversary can exploit the information obtained from the analysis of HDT($\mathbf{e}$, t) properties of APUF, XOR PUF and LSPUF to develop statistical attacks on them, if the ideal value of HDT($\mathbf{e}$, t) = 0.5 is not achieved for a given PUF. We validate our theoretical observations using the simulated and FPGA implemented APUF, XOR PUF and LSPUF designs.

Additional Key Words and Phrases: Arbiter physically unclonable function (APUF), adaptive chosen-challenge attack, hamming distance test, propagation criteria, statistical attack, unpredictability property.

1 INTRODUCTION

Physically Unclonable Functions (PUFs) are promising hardware security primitives, with numerous proposed applications which either complement or substitute traditional cryptographic algorithms [1, 4, 8, 25]. Many attacks have been proposed with varying degrees of success against PUFs, including physical attack [23], machine learning (ML) based *model building attacks* [3, 7, 19], and cryptanalysis [15, 20]. Acceptability of a given PUF variant usually involves compromise between its hardware footprint, statistical behavior, and robustness against attacks. For example, the Arbiter PUF (APUF) [7], a classic and widely studied PUF circuit based on process variation induced delay difference of signal propagation paths, has low hardware complexity and area, but is extremely susceptible to model building attacks. Hence, the APUF is rarely used as a standalone PUF, but is usually used as a component of more secure PUF circuits, e.g. Lightweight Secure

*This is a revised version of the article published in ACM TODAES, 2016 [16]. In [16], we reported an incorrect comparison between our work and result reported in [5, Fig. 14]. The initial idea of output transition probability of APUF and its relationship with Hamming distance of parity vectors (obtained from challenge) was presented by Delvaux *et al.* in [5], and this result has similarity with our reported result on this fact. Thus, our work should be considered as a follow-up work of [5].

†*Author's addresses: The authors are with the Secured Embedded Architecture Laboratory (SEAL), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, INDIA–721302.* E-mail:phuongha.ntu@gmail.com, dpsahoo.cs@gmail.com, rschakraborty@cse.iitkgp.ernet.in, debdeep@cse.iitkgp.ernet.in

PUF (LSPUF) [10], Composite PUF [21] and XOR PUF [22]. However, recently, even some of these "secure PUFs" have been attacked successfully [3, 14, 20].

One of the most important security properties of a PUF is the unpredictability of its challenge-response behavior. In essence, this means that an adversary should not be successful in finding a statistically significant relationship among CRPs of a PUF. However, the current published literature lacks a concrete methodology to quantitatively evaluate the unpredictability of a PUF. In [11], we find the notion of *Predictability Test Suite* that includes a list of PUF testing schemes to evaluate the unpredictablity. One such testing scheme is the *Hamming Distance Test* (denoted by HDT(t)), objective of which is to measure average output transition probability due to challenge pairs $(\mathbf{c}_i, \mathbf{c}_j)$ with Hamming distance[1] $\text{HD}(\mathbf{c}_i, \mathbf{c}_j) = t$. This test is a simplified case of *Propagation Criteria* (PC) [18], since the PC(t) test measures average output transition probability due to challenge pairs $(\mathbf{c}_i, \mathbf{c}_j)$ with $\text{HD}(\mathbf{c}_i, \mathbf{c}_j) \leq t$. Let us denote the mismatch pattern $\mathbf{e}$ between challenge pair $(\mathbf{c}_i, \mathbf{c}_j)$ as $\mathbf{e} = \mathbf{c}_i \oplus \mathbf{c}_j$. According to [11], PUFs should ideally satisfy $\text{HDT}(t) = 0.5$. But we observe that an APUF [7] still leaks information to an adversary even though it satisfies $\text{HDT}(t) = 0.5$ property, since there are many mismatch patterns $\mathbf{e}$ with either very high or low output transition probability and it results in average output transition probability of approximately 0.5, i.e., $\text{HDT}(t) \approx 0.5$ (cf. Section 3.4). Based on this observation, we propose a new testing scheme $\text{HDT}(\mathbf{e}, t)$ to ensure the unpredictability property of PUF in the context of statistical attacks. Objective of $\text{HDT}(\mathbf{e}, t)$ test is to measure output transition probability of PUF for a given mismatch pattern $\mathbf{e}$ with Hamming weight[2] t, i.e., $t = \text{HW}(\mathbf{e})$. In this work, we show that $\text{HDT}(\mathbf{e}, t)$ is superior to $\text{HDT}(t)$ and $\text{PC}(t)$ tests. More specifically, if a PUF design qualifies $\text{HDT}(\mathbf{e}, t)$ test, then it also satisfies $\text{HDT}(t)$ and $\text{PC}(t)$, but the reverse does not hold.

The $\text{HDT}(t)$ test has been studied for APUF, XOR APUF and LSPUF in [11, 13]. In [11, Fig. 11], the authors reported that $\text{HDT}(t)$ of the classic APUF is approximately 0.5 for most of the t values. They also observed similar $\text{HDT}(t)$ values for the XOR APUF, though the authors in [5] claimed that the result reported in [11] is not correct, and they provided new results (cf. [5, Fig. 14]) along with analytical formulation. Actually, both the results reported in [11, Fig. 11] and [5] are correct, and they discussed two different aspects. In [11, Fig. 11], the authors reported the relationship between output transition probability and Hamming distance of challenge pairs. On the other hand, in [5], the authors considered the Hamming distance between a pair of parity vectors obtained from a given pair of challenges. In this paper, we discuss both these facts based on the theoretical analyses.

We can also find consideration of the *Strict Avalanche Criteria* (SAC)[2], also known as HDT(1) and PC(1), property of APUF, XOR APUF, and APUF with its dependence on the input network of LSPUF in [11]. The reported result regarding SAC property in [11] is actually equivalent to our proposed $\text{HDT}(\mathbf{e}, t = 1)$ test, but they did not extend the study for arbitrary values of t. In this work, we provide a generic and analytical discussion on $\text{HDT}(\mathbf{e}, t)$ property of classic APUF, XOR APUF and LSPUF, and this work is motivated by the result reported in [5, Fig. 14]. Compared to [5, Fig. 14] where the authors provided analytic expressions of output transition probability for XOR APUF without detailed derivation, in this paper, we provide a generic expression and its detailed derivation which would be useful to the readers. In addition, we develop an adaptive

[1]The Hamming weight of a binary vector $\mathbf{a}$ is the number of 1's in $\mathbf{a}$, denoted by $\text{HW}(\mathbf{a})$. The Hamming distance between two binary vectors $\mathbf{a}$ and $\mathbf{b}$ of equal length is denoted by $\text{HD}(\mathbf{a}, \mathbf{b})$ and it is defined as: $\text{HD}(\mathbf{a}, \mathbf{b}) = \text{HW}(\mathbf{a} \oplus \mathbf{b})$, where $\oplus$ is Exclusive-OR operation.

[2]A PUF with n-bit challenge and 1-bit response is said to satisfy SAC if its response transition occurs with probability 0.5 whenever any one of the challenge bit is complemented.

chosen-challenge attack based on the poor HDT($\mathbf{e}, t$) property of above mentioned three APUF designs.

The major contributions of this paper are summarized as follows:

(1) We propose a new statistical test HDT($\mathbf{e}, t$) to observe the existence of a statistically significant relationship among the CRPs of a PUF instance. The existence of such a relationship reduces the unpredictability of a PUF. We also show that this test is superior than HDT(t) and PC(t) tests. As case studies, we provide a comprehensive analytical discussion on HDT($\mathbf{e}, t$) properties of the classic APUF, XOR APUF and LSPUF. In addition, we establish a relationship among HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties of these PUF designs.
(2) Based on HDT($\mathbf{e}, t$) test, we develop a (adaptive) chosen-challenge attack scheme for APUF, XOR APUF and LSPUF. Objective of this attack is to derive many related challenges from a set of known challenges, with high prediction accuracy for the corresponding responses. We also show that the LSPUF input network is not enough to withstand this attack, although the input network was proposed to resist this type of attack in the first place. To the best of our knowledge, this the first statistical attack on LSPUF based on its input network.
(3) We validate our theoretical analyses using 64-bit and 128-bit simulated and FPGA implemented APUFs, XOR APUFs and LSPUFs. An important conclusion from our experimental results is that HDT($\mathbf{e}, t$) values of FPGA implemented PUFs are poor compared to that of simulated PUFs. In addition, we observe that HDT($\mathbf{e}, t$) is poorer for larger PUF instances compared to its variants with smaller challenge.

Organization of the Paper. The rest of paper is organized as follows. In Section 2, we introduce a notation system that will be used in rest of the paper, along with basic security notion of PUF, adversary threat models and propagation criteria. Section 3 discusses HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties of APUF. We extend the analysis of HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties for XOR APUF and LSPUF in Sections 4 and 5, respectively. Various security threats on APUF and its variants based on HDT($\mathbf{e}, t$) are discussed in Section 6. Section 7 provides simulation and experimental results. Finally, concluding comments are provided in Section 8.

2 PRELIMINARIES

2.1 Notations

We use following notation system in the rest of the paper. A letter in bold font refers to a vector, e.g $\mathbf{a}$. A vector with m-components is represented as $\mathbf{a} = (\mathbf{a}[0], \ldots, \mathbf{a}[i], \ldots, \mathbf{a}[m-1])$, where $\mathbf{a}[i]$ is the ith component of the vector. We use $\mathbf{a}[i{:}j]$ to denote a sub-vector $(\mathbf{a}[i], \ldots, \mathbf{a}[j])$. The transpose of a vector $\mathbf{a}$ is denoted by $\mathbf{a}^{\mathsf{T}}$. A scalar is denoted by a lower-case letter, e.g. n. In $a = b\%c$, $\%$ denotes the modulo operation and a is the remainder of the division b/c. The right rotation of a binary vector $\mathbf{a}$ by k positions is denoted by $\mathbf{a} \ggg k$. A set is represented by calligraphic font, e.g. set $\mathcal{D}$ and its cardinality is denoted as $|\mathcal{D}|$. We denote random variables by upper-case letters, e.g. X. $\Pr(X = x)$ is used to denote the probability of the event $X = x$, and $\mathrm{E}[X]$, σ_X and $\mathrm{Var}(X)$ (or σ_X^2) are used to denote mean, standard deviation and variance of the random variable X, respectively. Random variable X following a Gaussian probability distribution function is denoted as $X \sim \mathcal{N}(\mu, \sigma^2)$. $\phi_{\mu,\sigma^2}()$ and $\Phi_{\mu,\sigma^2}()$ represent the probability density function (PDF) and cumulative distribution function (CDF) of Gaussian random variable, respectively; $\phi()$ and $\Phi()$ represent the PDF and CDF of standard normal random variable, respectively. Hamming distance between two binary vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by HD($\mathbf{a}, \mathbf{b}$), and Hamming weight of $\mathbf{a}$ is denoted by HW($\mathbf{a}$).

2.2 Security Notion of PUF

Although PUF is physically unclonable, it is not enough to prevent different types of protocol level attacks, like attacks on remote authentication of hardware and remote activation of IP licenses [5]. In these cases, it does not matter whether an adversary is impersonating a PUF in physical or non-physical ways. Adversary can build a mathematical model based on the eavesdropped CRPs to approximate the behavior of a PUF very closely in an efficient way. It has been observed in the PUF literature that researchers are interested in practical attacks, i.e., prediction success probability of a model approaches 1. Whereas in traditional cryptography, any attack technique that achieves a prediction success probability non-negligible[3] greater than 0.5 (random guess) is considered as significant threat, such is not the traditional viewpoint in research on PUF security. For example, in [24], authors reported the proposed design to be resistant against modeling attacks even though an adversary can build a model for the PUF design with prediction success probability 0.90. In this context, we prefer to introduce two different definitions of predictability with respect to the prediction accuracy to avoid the confusion regarding secure PUF: *Strong Predictability* and *Weak Predictability*.

(1) **Strong Predictability**: The responses of a PUF are said to be *strongly predictable* if an adversary can build a model in polynomial (in challenge size) time and data complexities with prediction accuracy that approaches the reliability of the PUF. For instance, an APUF design is considered as strongly predictable [7], as an adversary can build a ML-based model with prediction accuracy approximately 99% while reliability of APUF is 100%.
(2) **Weak Predictability**: The responses are *weakly predictable* if the prediction accuracy of model is non-negligible better than that of random guess, in polynomial (in challenge size) time and data complexities. Precisely, the prediction accuracy of a weak model with 1-bit response is $\frac{1}{2} + \delta$, where δ is non-negligible.

The reader might think why we relate the prediction accuracy with the reliability of a PUF in case of the strong predictability. We have introduced the notion of strong predictability to imply that if an adversary can build a strong PUF model, then she can impersonate the physical PUF. A successful impersonation means the PUF and its model are indistinguishable. Since there are some error in physical PUF (denoted by ϵ_{P}) due to the lack of perfect reliability, and error in the model (denoted by ϵ_{M}) due the failure in convergence of the modeling algorithm, the verifier can employ ϵ_{P} and ϵ_{M} to distinguish the PUF from its model while $|\epsilon_{\mathrm{P}} - \epsilon_{\mathrm{M}}|$ is not negligible. In other words, if $|\epsilon_{\mathrm{P}} - \epsilon_{\mathrm{M}}|$ is not negligible, then PUF model is weak and it cannot be used to impersonate the PUF.

For a PUF design, existence of either the weak or strong prediction models implies that PUF is not secure by design, but still it might be used in a secure way in some security protocols. The following definition formalizes the notion of a truly secure PUF [17, 20]:

Definition 1 (**Truly Secure PUF Design**). *A PUF instance with n-bit challenge* $\mathbf{c}$ *and m-bit response* $\mathbf{r}$ *is considered to be* secure *if and only if there is no adversary who can predict the responses with probability non-negligible greater than* $1/2^m$ *(random guess probability) with following computing constraints: (1) time complexity of attack is less than* 2^n *queries of PUF, and (2) data complexity, i.e., the number of challenge-response pairs (CRPs) required to predict the response of a given challenge, is less than* 2^n.

2.3 Adversary Threat Models

In the context of PUFs, we consider the following two adversary models:

[3] A function g is **negligible** [6] if for all positive constant b, there is an N_b and for all $x > N_b$ it satisfies that $g(x) < \frac{1}{x^b}$.

- **Chosen-challenge Adversary** (CCA): In *Learning phase*, a chosen-challenge adversary can query the PUF oracle $\mathcal{P}$ for response to an arbitrary challenge $\mathbf{c} \in Q$ that she chooses, and $|Q|$ is polynomially bounded. After this phase, the adversary cannot query the PUF oracle. In *Attack phase*, the adversary predicts the response to a challenge $\mathbf{c}_{\text{queried}} \notin Q$ based on the responses of challenges in Q.
- **Adaptive Chosen-challenge Adversary** (ACCA): In the learning phase of this model, the adversary, depending on the challenge $\mathbf{c}_{\text{queried}}$ for which she is required to predict the response, intelligently (or adaptively) chooses a set of challenges Q provided that $\mathbf{c}_{\text{queried}} \notin Q$. The adversary is allowed to access the oracle $\mathcal{P}$ to get responses to the challenges in Q. In attack phase, the adversary uses the responses of this adaptive challenge set Q to predict the response for challenge $\mathbf{c}_{\text{queried}}$. This type of attack model is developed based on the following fact: for a secure PUF, the information of any CRP should not leak the information about other CRPs, i.e., there should not exist any *related* CRPs.

For PUF modeling attacks, we usually consider a CCA threat model. The CCA model is more practical in the context of PUFs, e.g. in a scenario where an adversary eavesdrops the CRPs being used for authentication, and subsequently uses them to build a PUF model, without getting actual physical access to the PUF. The adversary model ACCA is stronger than CCA. However, we consider the ACCA model so that the designer can evaluate the security of the PUF, and ensure that the PUF is secure under the ACCA attack scenario.

In this paper later we use the ACCA model to define the *related challenges* of a PUF instance, and information leakage about responses of unseen challenges.

2.4 Propagation Criteria

Definition 2 (Propagation Criteria). *An n-bit input, 1-bit output Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is said to satisfy propagation criteria of degree t (*PC(t), $1 \leq t \leq n$*), if the value of the function f complements with probability of one-half, whenever t or less number of bits of the input are complemented [18]. This can be stated more formally as follows:*

$$\Pr[(f(\mathbf{c}) \oplus f(\mathbf{c} \oplus \mathbf{e})) = 1] = \frac{1}{2}, \tag{1}$$

where $\mathbf{c}, \mathbf{e} \in \{0,1\}^n$, and HW($\mathbf{e}$) $\leq t$. *Typically, each $\mathbf{e}$ defines an input-bit flipping pattern. Note that the strict avalanche criterion (SAC) is equivalent to* PC(1).

In [11], the authors proposed the concept of *predictability test* of PUF design to measure the average output transition probability due to challenge pairs with a given Hamming distance between them. This test is called as *Hamming Distance Test*, and denoted by HDT(t). Formally, it is said that a PUF satisfies the HDT(t) test when:

$$\text{HDT}(t) = \Pr_{\mathbf{c},\mathbf{e}}[(f(\mathbf{c}) \oplus f(\mathbf{c} \oplus \mathbf{e})) = 1] = \frac{1}{2}, \tag{2}$$

$\mathbf{c}, \mathbf{e} \in \{0,1\}^n$, and HW($\mathbf{e}$) = t. HDT(t) test can be considered as a simplified version of PC(t) test. The PC(t) test is not discussed in [11], and we show that PC(t) value of a PUF design can be 0.5 even if the PUF design does not satisfy HDT(t) = 0.5, as:

$$\text{PC}(t) = \frac{1}{t} \sum_{i=1}^{t} \text{HDT}(i). \tag{3}$$

We introduce a new *Hamming Distance Test*, where we consider both the mismatch pattern vector $\mathbf{e}$ and its Hamming weight t = HW($\mathbf{e}$). We denote this test by HDT($\mathbf{e}, t$), and it is formally

defined as follows. A PUF is said to satisfy the HDT($\mathbf{e}, t$) if for all pairs of ($\mathbf{e}, t$):

$$\text{HDT}(\mathbf{e}, t) = \Pr_{\mathbf{c}}[(f(\mathbf{c}) \oplus f(\mathbf{c} \oplus \mathbf{e})) = 1] = \frac{1}{2}, \tag{4}$$

where $\mathbf{c}, \mathbf{e} \in \{0,1\}^n$, and $t = \text{HW}(\mathbf{e})$. Let $\mathcal{E}_t$ be a set of all the pattern vectors $\mathbf{e}$ with $t = \text{HW}(\mathbf{e})$, then the relationship between HDT(t) and HDT($\mathbf{e}, t$) is:

$$\text{HDT}(t) = \frac{1}{|\mathcal{E}_t|} \sum_{\mathbf{e} \in \mathcal{E}_t} \text{HDT}(\mathbf{e}, t), \tag{5}$$

where $|\mathcal{E}_t|$ is the cardinality of set $\mathcal{E}_t$. This implies that value of HDT(t) can be 0.5 even though there exist a few patterns for which HDT($\mathbf{e}, t$) is either approximately 0 or 1. Thus, to ensure the security of a PUF design under ACCA adversary, we need to show that HDT($\mathbf{e}, t$) $\approx$ 0.5 for all values of $\mathbf{e}$ and t.

In this paper, we discuss the HDT($\mathbf{e}, t$) property to evaluate the unpredictability of APUF, XOR APUF and LSPUF designs. The HDT($\mathbf{e}, t$) test is a generalization of the HDT(t) and PC(t) tests – if a PUF satisfies the HDT($\mathbf{e}, t$) test, then it also satisfies the HDT(t) and PC(t) tests.

3 ARBITER PUF AND ITS HDT($\mathbf{e}, t$), HDT(t) AND PC(t) PROPERTIES

3.1 Arbiter PUF Design and Its Linear Additive Delay Model

The Arbiter PUF (APUF) introduced in [7] is depicted in Fig. 1. It is a delay-based silicon PUF that exploits random process variation effects in silicon, in terms of the delay difference of two symmetrically laid out parallel paths. Typically, the response (r) of an APUF is defined by Eq. (6).

$$r = \begin{cases} 1, & \text{if the delay of the signal at the upper input of the arbiter is smaller} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

However, as mentioned previously, the APUF is known to be vulnerable to machine learning based modeling attack (MLMA) [7, 19]. In [7], a *linear additive delay model* of the APUF was derived. Let Δ be the delay difference between delays of the top and bottom paths of an n-bit APUF, for a given challenge, and it is modeled as [7]:

$$\Delta = \mathbf{w}[0]\mathbf{\Phi}[0] + \cdots + \mathbf{w}[j]\mathbf{\Phi}[j] + \cdots + \mathbf{w}[n]\mathbf{\Phi}[n] = \mathbf{w}^{\mathsf{T}}\mathbf{\Phi}. \tag{7}$$

The vector $\mathbf{w}$ is defined based on the delays of APUF's delay components, and complete definition of the vector $\mathbf{w}$ can be found in [7]. The vector $\mathbf{\Phi}$ is derived from the input challenge $\mathbf{c}$, and is defined as follows:

$$\mathbf{\Phi}[j] = \begin{cases} \prod_{k=j}^{n-1}(1 - 2\mathbf{c}[k]), & j = 0, \ldots, n-1 \\ 1, & j = n. \end{cases} \tag{8}$$

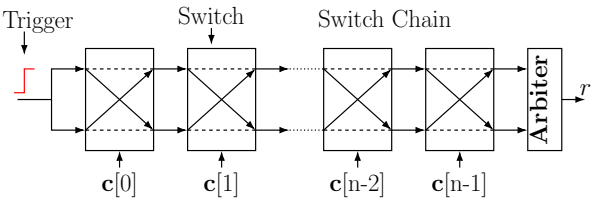
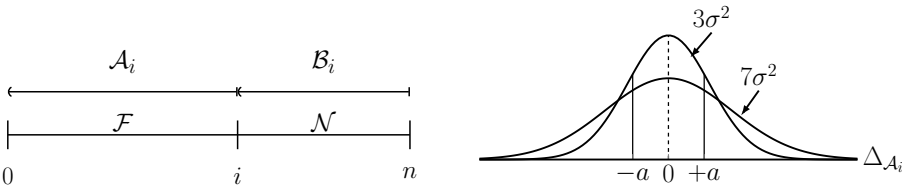


Fig. 1. Arbiter PUF.



(a) *Flipping zone* $\mathcal{F}$ and *non-flipping zone* $\mathcal{N}$ in the index range of Φ due to flip in i-th bit of challenge $\mathbf{c}$.

(b) Normal distributions with $\mu = 0$, and variances $3\sigma^2$ and $7\sigma^2$. Probability $\Pr(|\Delta_{\mathcal{A}_i}| > a)$ is larger for $\mathcal{N}(0, 7\sigma^2)$ than that of $\mathcal{N}(0, 3\sigma^2)$.

Fig. 2. A few aspects of HDT($\mathbf{e}$, 1) computation.

3.2 Computation of HDT($\mathbf{e}$, 1) of APUF

First, we discuss HDT($\mathbf{e}$, 1) property of APUF, which is reported in [10, 12] as SAC property. Let us use the notation $\mathbf{c}[i] \in \Phi[j]$ to state the fact that $\mathbf{c}[i]$ is present in the expression of $\Phi[j]$ (cf. Eq. (8)), and otherwise, we write $\mathbf{c}[i] \notin \Phi[j]$. Without loss of generality and for the sake of analysis, we assume that all components of $\mathbf{w}$ are independent and identically distributed random variables, and each component $\mathbf{w}[i]$ follows a Gaussian distribution [7] with mean $\mu = 0$, and variance σ^2, i.e., $\mathbf{w}[i] \sim \mathcal{N}(0, \sigma^2)$, $i = 0, \ldots, n$. Let us define following two sets representing the partitions of indices $[0, n]$ of Φ with respect to $\mathbf{c}[i]$, $i = 0, \ldots, n-1$,

$$\mathcal{A}_i = \{j : \mathbf{c}[i] \in \Phi[j]\} \qquad \text{and} \qquad \mathcal{B}_i = \{j : \mathbf{c}[i] \notin \Phi[j]\}$$

From Eq. (8) and Fig. 2a, we have: $\mathcal{A}_i = \{0, 1, \ldots, i\}$ and $\mathcal{B}_i = \{i+1, \ldots, n\}$. In addition, we define two delay values corresponding to $\mathcal{A}_i$ and $\mathcal{B}_i$ as: $\Delta_{\mathcal{A}_i} = \sum_{j \in \mathcal{A}_i} \mathbf{w}[j]\Phi[j]$ and $\Delta_{\mathcal{B}_i} = \sum_{j \in \mathcal{B}_i} \mathbf{w}[j]\Phi[j]$. Then $\Delta = \Delta_{\mathcal{A}_i} + \Delta_{\mathcal{B}_i}$.

It can be observed that signs of all $\Phi[j], j \in \mathcal{A}_i$ would be changed if $\mathbf{c}[i]$ flips, while signs of all $\Phi[j]$ for $j \in \mathcal{B}_i$ would not be changed even if $\mathbf{c}[i]$ flips. In Fig. 2a, the $\mathcal{A}_i$ and $\mathcal{B}_i$ of challenge bit $\mathbf{c}[i]$ are depicted; these two sets partition indices of Φ into two regions: i) *flipping zone* $\mathcal{F}$ where signs of all $\Phi[j]$ flip, and ii) *non-flipping zone* $\mathcal{N}$ where signs of all $\Phi[j]$ remain unchanged.

Since $\Delta_{\mathcal{B}_i}$ does not depend on the value of $\mathbf{c}[i]$, the sign of Δ is changed when $\mathbf{c}[i]$ flips and $|\Delta_{\mathcal{A}_i}| > |\Delta_{\mathcal{B}_i}|$. In other words, the response bit r flips when $\mathbf{c}[i]$ flips and $|\Delta_{\mathcal{A}_i}| > |\Delta_{\mathcal{B}_i}|$. Let X_i be a random variable which is defined as follows:

$$X_i = \begin{cases} 1, & |\Delta_{\mathcal{A}_i}| > |\Delta_{\mathcal{B}_i}| \\ 0, & \text{otherwise.} \end{cases}$$

Let us denote $\mathbf{e}_i$ to be a binary vector where only the component $\mathbf{e}[i] = 1$. Let r be the response of APUF to challenge $\mathbf{c}$, then HDT($\mathbf{e}_i$, 1) $= \Pr(X_i = 1) = \Pr(\bar{r} \mid \bar{\mathbf{c}}[i])$, where $\bar{r} = r \oplus 1$ and $\bar{\mathbf{c}}[i] = \mathbf{c}[i] \oplus 1$. Thus, $X_i = 1$ denotes the event that the output bit flips when the ith challenge bit flips.

It is well-known that the sum of mutually independent Gaussian random variables also follows a Gaussian distribution, and thus if $W_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, then $W = \sum_i W_i$ and $W \sim \mathcal{N}(\sum_i \mu_i, \sum_i \sigma_i^2)$. Hence,

$$\Delta_{\mathcal{A}_i} \sim \mathcal{N}(0, (i+1)\sigma^2) \qquad \text{and} \qquad \Delta_{\mathcal{B}_i} \sim \mathcal{N}(0, (n-i)\sigma^2). \tag{9}$$

From Eq. (9), it is evident that if $(i+1) \gg (n-i)$ (or $(i+1) \ll (n-i)$), then the event $|\Delta_{\mathcal{A}_i}| > |\Delta_{\mathcal{B}_i}|$ (output r flips) can occur with a high probability (or a low probability), and then $\Pr(X_i = 1) \gg 0.5$ (or $\Pr(X_i = 1) \ll 0.5$).

To visualize this property, we depict the PDFs of normal distributions for the following cases: $|\mathcal{A}_i| = 3$ and $|\mathcal{A}_i| = 7$ in Fig. 2b. It can be observed from Fig. 2b that increasing value of $|\mathcal{A}_i|$ results in increment of $\Pr(|\Delta_{\mathcal{A}_i}| > a)$, where $a > 0$. Moreover, if we increase i, then $(i+1)$ increases and $(n-i)$ decreases. Thus, as we increase i from 0 to n, then $\Pr(X_i = 1) = \Pr(|\Delta_{\mathcal{A}_i}| > |\Delta_{\mathcal{B}_i}|)$ increases from 0 to 1. As observed in [10, 12], for a given i such that $(i+1) \approx (n-i)$, the output r flips with a probability approximately 0.5, i.e., $\Pr(X_i = 1) \approx 0.5$.

3.3 Computation of HDT$(\mathbf{e}, t)$ of APUF

In this section, we generalize the above concept by focusing on HDT$(\mathbf{e}, t)$ property with $1 < t \leq n$. We assume that t bits $\mathbf{c}[i_1], \ldots, \mathbf{c}[i_t]$ are flipped, $0 \leq i_1 < \cdots < i_t \leq n-1, 1 \leq t \leq n$. We have the following important observation:

Observation 1. *Given $\Phi[j] = \prod_{k=j}^{n-1}(1 - 2\mathbf{c}[k])$, $j = 0, \ldots, n-1$, it is observed that if the number of flipping bits which are present in the expression of $\Phi[j]$ is even, then the sign of $\Phi[j]$ does not change. Otherwise, the sign of $\Phi[j]$ flips. Note that $\Phi[n] = 1$, and thus, there will be no flip in its sign.*

We define following two sets based on the indices of $\Phi[0, n]$:

$$\mathcal{A}_{i_1 i_2 \ldots i_t} = \{j : \text{the sign of } \Phi[j] \text{ flips when all } \mathbf{c}[i_1], \ldots, \mathbf{c}[i_t] \text{ flip}\}$$
$$\mathcal{B}_{i_1 i_2 \ldots i_t} = \{j : \text{the sign of } \Phi[j] \text{ does not flip when all } \mathbf{c}[i_1], \ldots, \mathbf{c}[i_t] \text{ flip}\}.$$

In addition, we define two delay values corresponding to $\mathcal{A}_{i_1 i_2 \ldots i_t}$ and $\mathcal{B}_{i_1 i_2 \ldots i_t}$ as:

$$\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} = \sum_{j \in \mathcal{A}_{i_1 i_2 \ldots i_t}} \mathbf{w}[j]\Phi[j] \quad \text{and} \quad \Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}} = \sum_{j \in \mathcal{B}_{i_1 i_2 \ldots i_t}} \mathbf{w}[j]\Phi[j].$$

Then $\Delta = \Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} + \Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}}$.

Let $X_{i_1 i_2 \ldots i_t}$ be a random variable which is defined as follows:

$$X_{i_1 i_2 \ldots i_t} = \begin{cases} 1, & |\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}}| > |\Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}}| \\ 0, & \text{otherwise.} \end{cases}$$

Let us denote $\mathbf{e}_{i_1, \ldots, i_t}$ as a binary vector with only components $\mathbf{e}[i_1] = \cdots = \mathbf{e}[i_t] = 1$. Let r be the response of APUF to challenge $\mathbf{c}$, then

$$\text{HDT}(\mathbf{e}_{i_1, \ldots, i_t}, t) = \Pr(X_{i_1 i_2 \ldots i_t} = 1) = \Pr\left(\bar{r} \mid \bar{\mathbf{c}}[i_1], \bar{\mathbf{c}}[i_2], \ldots, \bar{\mathbf{c}}[i_t]\right), \tag{10}$$

where $\bar{r} = r \oplus 1$ and $\bar{\mathbf{c}}[i_k] = \mathbf{c}[i_k] \oplus 1, 1 \leq k \leq t$. Based on the assumption we made for Eq. (9) regarding the sum of Gaussian random variables, we have:

$$\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} \sim \mathcal{N}(0, |\mathcal{A}_{i_1 i_2 \ldots i_t}|\sigma^2) \quad \text{and} \quad \Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}} \sim \mathcal{N}(0, |\mathcal{B}_{i_1 i_2 \ldots i_t}|\sigma^2). \tag{11}$$

In Appendix A, we discuss an analytical expression for HDT$(\mathbf{e}, t)$ of an APUF. In practice, computation of HDT$(\mathbf{e}, t)$ by evaluating analytical expression might not be feasible because the value of σ (standard deviation of delay distribution of delay components) might be unknown. Instead, we provide an alternative approach to estimate HDT$(\mathbf{e}, t)$ by directly using CRPs of a PUF instance as described in Algorithm 1. Algorithm 1 uses N random challenges, and for each challenge $\mathbf{c}$ (Line 2), it generates another challenge $\hat{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e}_{i_1, \ldots, i_t}$ using pattern vector $\mathbf{e}_{i_1, \ldots, i_t}$ (Line 5). If responses r and $\hat{r}$ due to $\mathbf{c}$ and $\hat{\mathbf{c}}$, respectively, are not equal, then the algorithm keeps track of this event in the variable *count* (Line 8). Finally, HDT$(\mathbf{e}_{i_1, \ldots, i_t}, t)$ is computed as a ratio of *count* to N (Line 11). The accuracy of computed HDT$(\mathbf{e}_{i_1, \ldots, i_t}, t)$ value depends on the parameter N, and large N value implies better accuracy.

Algorithm 1 Computation of HDT($\mathbf{e}_{i_1,\ldots,i_t}, t$) of an APUF instance w.r.t. t flipping bits

Input: P is an APUF instance with n-bit challenge; binary pattern vector $\mathbf{e}_{i_1,\ldots,i_t}$ where $\mathbf{e}[j] = 1$ only for $j \in \{i_1, \ldots, i_t\}$, and $\mathbf{e}[j] = 0$ otherwise; N is the number of CRPs
Output: Value of HDT($\mathbf{e}_{i_1,\ldots,i_t}, t$)

```
count ← 0
for k = 1 to N do
    c ← a randomly generated challenge
    r ← P(c) {Evaluate APUF with c}
    ĉ ← c ⊕ e_{i_1,...,i_t} {Modified challenge ĉ}
    r̂ ← P(ĉ) {Evaluate APUF with ĉ}
    if r ≠ r̂ then
        count ← count + 1
    end if
end for
HDT(e_{i_1,...,i_t}, t) ← count/N
```

3.4 HDT(t) and PC(t) Properties of APUF

In the context of APUF, we can observe a few interesting properties of HDT(t) and PC(t). These properties are previously reported in [11] with the help of experimental result, but the authors did not explain their observations from a theoretical viewpoint. We now state and prove these properties in the following theorem:

Theorem 1. *The* HDT(t) *and* PC(t) *properties of an APUF depend on the parity of* t *in the following way:*

(1) HDT(t)*: (a) For odd* t, HDT(t) *is equal to* 0.5*, and (b) for even* t, HDT(t) *is less than* 0.5*, and it approaches* 0.5 *with increasing value of* t.
(2) PC(t)*: for all* t, PC(t) *is less than 0.5, and it approaches* 0.5 *with increasing value of* t.

Proof. The proof of this theorem is provided in Appendix B. □

3.5 Discussion on Related Works

In [11], the authors proposed HDT(t) test and according to their reported experimental results, the value of HDT(t) is approximately 0.5 for most of the t values (cf. [11, Fig. 11]). Our analysis also results to a similar trend for HDT(t), and in addition, we have provided an analytical expression for HDT(t) property of APUF. Our analysis of HDT($\mathbf{e}, t$) also matches with the result in [5, Fig. 14].

4 x-XOR APUF AND ITS HDT($\mathbf{e}, t$), HDT(t) AND PC(t) PROPERTIES

4.1 Design Overview

As described in [22], an x-XOR APUF (cf. Fig. 3) is constructed based on a set of x n-bit APUF instances $A_0, \ldots, A_{x-1}$. For a given challenge $\mathbf{c} = (\mathbf{c}[0], \ldots, \mathbf{c}[n-1])$, the response r is generated as follows: $r = r_0 \oplus \cdots \oplus r_i \oplus \cdots \oplus r_{x-1}$, where $r_i = A_i(\mathbf{c})$, $i = 0, \ldots, x-1$.

Next, we discuss HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties of XOR APUF.

4.2 Computation of HDT(e, t) of x-XOR APUF

Let $Y_{i_1 i_2 \ldots i_t}$ be a random variable defined as follows:

$$Y_{i_1 i_2 \ldots i_t} = \begin{cases} 1, & r \text{ flips when bits } \mathbf{c}[i_1], \ldots, \mathbf{c}[i_t] \text{ flip, where } 0 \le i_1 < \cdots < i_t \le n-1 \\ 0, & \text{otherwise.} \end{cases}$$

Then, $\mathrm{HDT}(\mathbf{e}, t) = \Pr_{\mathbf{c}}(Y_{i_1 i_2 \ldots i_t} = 1) = \Pr_{\mathbf{c}}(\bar{r} \mid \bar{\mathbf{c}}[i_1], \bar{\mathbf{c}}[i_2], \ldots, \bar{\mathbf{c}}[i_t])$, where $\bar{r} = r \oplus 1$ and $\bar{\mathbf{c}}[i_k] = \mathbf{c}[i_k] \oplus 1, 1 \le k \le t$. To simplify the analysis, we assume that all APUFs $A_0, \ldots, A_{x-1}$ are mutually independent (this assumption might not be entirely correct in actual implementations). Since the challenge $\mathbf{c}$ is the input to all APUFs $A_0, \ldots, A_{x-1}$ and all the outputs $r_0, \ldots, r_{x-1}$ are mutually independent, we can consider that $\Pr(\bar{r}_i \mid \bar{\mathbf{c}}[i_1], \ldots, \bar{\mathbf{c}}[i_t])$ of each APUF $A_i, i = 0, \ldots, x-1$, is same, and equal to $p_{i_1 \ldots i_t}$ (see Section 3), where: $p_{i_1 \ldots i_t} = \Pr(\bar{r}_i \mid \bar{\mathbf{c}}[i_1], \ldots, \bar{\mathbf{c}}[i_t])$.

Since the output of x-XOR APUF flips when k APUF outputs (out of x APUF outputs) flip and k is odd, $\mathrm{HDT}(\mathbf{e}, t)$ of x-XOR APUF can be computed as described in the following theorem:

Theorem 2. *Let us define $p = p_{i_1 \ldots i_t}$, then*

$$\Pr(Y_{i_1 i_2 \ldots i_t} = 1) = \sum_{k=1, k\%2=1}^{x} \binom{x}{k} p^k (1-p)^{x-k} = \frac{1 - (1-2p)^x}{2}. \tag{12}$$

For example, in case of 2-XOR APUF, the $\Pr(Y_{i_1 i_2 \ldots i_t} = 1) = 2p(p-1)$ and for 3-XOR APUF, the $\Pr(Y_{i_1 i_2 \ldots i_t} = 1) = 3p(1-p)^2 + p^3$.

4.3 Computations of HDT(t) and PC(t) of XOR APUFs

In this section, we discuss $\mathrm{HDT}(t)$ and $\mathrm{PC}(t)$ properties of XOR APUF. The main results can be summarized by the following theorem:

Theorem 3. *The $\mathrm{HDT}(t)$ and $\mathrm{PC}(t)$ properties of an XOR APUF are:*

(1) $\mathrm{HDT}(t)$ *is approximately* 0.5 *for all t. Particularly,* $\mathrm{HDT}(t)$ *is equal to* 0.5 *for odd value of t, and for even t,* $\mathrm{HDT}(t)$ *approaches to* 0.5 *with increasing t.*
(2) $\mathrm{PC}(t)$ *is approximately* 0.5 *for all t.*

Proof. Let $r = r_0 \oplus r_1 \oplus \cdots \oplus r_{x-1}$ be the response of x-XOR APUF to challenge $\mathbf{c}$ and $r_i, i \in [1 : x-1]$ be the response of APUF A_i. Let us denote $r' = r_1 \oplus \cdots \oplus r_{x-1}$, and then output r of an x-XOR APUF can be rewritten as $r = r_0 \oplus r'$. Let X, Y and Z be random variables representing the following events:

(1) $X = \begin{cases} 1, & \text{if } r_0 \text{ flips due to flipping pattern } \mathbf{e} \text{ with challenge } \mathbf{c} \text{ and } \mathrm{HW}(\mathbf{e}) = t \\ 0, & \text{otherwise.} \end{cases}$
Indeed, $\Pr(X = 1)$ is the $\mathrm{HDT}(t)$ of APUF A_0 (see Theorem 1).

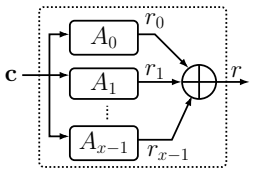


Fig. 3. x-XOR APUF.

(2) $Y = \begin{cases} 1, & \text{if } r' \text{ flips due to flipping pattern } \mathbf{e} \text{ with challenge } \mathbf{c} \text{ and } \mathrm{HW}(\mathbf{e}) = t \\ 0, & \text{otherwise.} \end{cases}$

The probability $\Pr(Y = 1)$ is the HDT(t) of $(x-1)$-XOR APUF constructed using APUFs $A_1, \ldots, A_{x-1}$.

(3) $Z = \begin{cases} 1, & \text{if } r \text{ flips due to flipping pattern } \mathbf{e} \text{ with challenge } \mathbf{c} \text{ and } \mathrm{HW}(\mathbf{e}) = t \\ 0, & \text{otherwise.} \end{cases}$

The probability $\Pr(Z = 1)$ is the HDT(t) of x-XOR APUF.

Note that r flips while r_0 flips and r' does not flip, or vice versa. Thus HDT(t) of x-XOR APUF will be:

$$\mathrm{HDT}(t) = \Pr(Z = 1) = \Pr(X = 1) \times \Pr(Y = 0) + \Pr(X = 0) \times \Pr(Y = 1). \tag{13}$$

From Theorem 1, we assume that $\mathrm{HDT}(t) \approx 0.5$ for APUF A_0 for all t (i.e., $\Pr(X = 1) = \Pr(X = 0) = 1/2$), though for even value of t, HDT(t) approaches 0.5 with increasing value of t. Based on this assumption, we can rewrite Eq. (13) as:

$$\begin{aligned} \Pr(Z = 1) &= \Pr(X = 1) \times \Pr(Y = 0) + \Pr(X = 0) \times \Pr(Y = 1) \\ &\approx 1/2 \times \Pr(Y = 0) + 1/2 \times \Pr(Y = 1) \\ &= 1/2 \times [\Pr(Y = 0) + \Pr(Y = 1)] = 1/2. \end{aligned} \tag{14}$$

Since $\mathrm{PC}(t) = 1/n \times \sum_{t=1}^{n} \mathrm{HDT}(t)$, the PC($t$) of XOR APUF is approximately 0.5 for all t. □

The experimental results, which are reported in [13, Fig. 4], match with our theoretical finding for HDT(t) of XOR APUF.

4.4 Discussion on Related Works

In [13, Fig. 4], HDT(t) of XOR APUF was experimentally computed, and shown to be 0.5, for $t = 1, \ldots, n$. However, in this paper, we have theoretically proved this with some additional information: the $\mathrm{HDT}(t) = 0.5$ for odd value of t, and for even t, HDT(t) approaches 0.5 with increasing t. Our HDT($\mathbf{e}, t$) test result has similarity with the result in [5, Fig. 14], and the only difference is that the results (conveying the same information) are reported in two different ways.

5 LSPUF AND ITS HDT($\mathbf{e}, t$), HDT(t) AND PC(t) PROPERTIES

5.1 Design Overview

The poor HDT($\mathbf{e}$, 1) properties of APUF and XOR APUF were the main motivation behind the design of LSPUF [10]. The architectural overview of LSPUF is shown in Fig. 4. An (n, m, k, x, s)-LSPUF

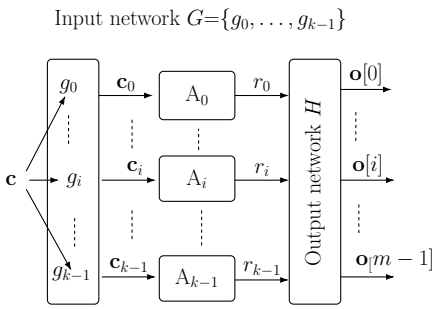


Fig. 4. Architectural overview of LSPUF.

instance consists of three layers: (1) *input network* $G = (g_0, \ldots, g_{k-1})$, (2) *PUF layer* consisting of k APUFs instances $A_0, \ldots, A_{k-1}$, and (3) *output network* H with m-bit output $\mathbf{o} = (\mathbf{o}[0], \ldots, \mathbf{o}[m-1])$. Details of these layers are discussed below.

5.1.1 Input Network G. The input network $G = (g_0, \ldots, g_{k-1})$ produces intermediate challenges $\mathbf{c}_0, \ldots, \mathbf{c}_{k-1}$ for k APUF instances in the PUF layer from the external input $\mathbf{c}$. For even value of n, mapping of $\mathbf{c}_i = g_i(\mathbf{c})$ are as follows:

(1) $\mathbf{d}_i = \mathbf{c} \ggg i$, where $\mathbf{c} \ggg i$ denotes the right rotation of challenge $\mathbf{c}$ by i positions, and $\mathbf{d}_i = (\mathbf{d}_i[0], \ldots, \mathbf{d}_i[n-1])$ and $\mathbf{c} = (\mathbf{c}[0], \ldots, \mathbf{c}[n-1])$.
(2) Let us denote $\mathbf{c}_i = (\mathbf{c}_i[0], \ldots, \mathbf{c}_i[n-1])$. Transformation of $\mathbf{d}_i$ to $\mathbf{c}_i$ is defined as:

$$\mathbf{c}_i\left[\frac{n}{2}\right] = \mathbf{d}_i[0]; \qquad \mathbf{c}_i\left[\frac{u}{2}\right] = \mathbf{d}_i[u] \oplus \mathbf{d}_i[u+1], u = 0, 2, \ldots, n-2; \tag{15}$$
$$\mathbf{c}_i\left[\frac{n+u+1}{2}\right] = \mathbf{d}_i[u] \oplus \mathbf{d}_i[u+1], u = 1, 3, \ldots, n-3$$

The main objective of the input network is to ensure that HDT($\mathbf{e}$, 1) value of LSPUF is around 0.5. Later, we show that this input network does not satisfy HDT($\mathbf{e}, t$) property in general, and the input network needs to be modified to improve the unpredictability property of LSPUF in the context of statistical attacks.

5.1.2 PUF Layer. This layer consists of a group of APUFs: $A_0, \ldots, A_{k-1}$. The input $\mathbf{c}_i$ to A_i is defined by $\mathbf{c}_i = g_i(\mathbf{c})$ for $i = 0, \ldots, k-1$, and A_i produces 1-bit response r_i. Thus this layer produces k-bit responses $(r_0, \ldots, r_{k-1})$, where $r_i = A_i(\mathbf{c}_i)$.

5.1.3 Output Network H. Finally, m-bit response $\mathbf{o} = (\mathbf{o}[0], \ldots, \mathbf{o}[m-1])$ is generated by the output network using the k-bit intermediate responses $(r_0, \ldots, r_{k-1})$ as: $\mathbf{o}[i] = \bigoplus_{j=0}^{x-1} r_{((i+s+j) \mod k)}$, where $i = 0, \ldots, m-1$, and x ($< k$) and s are security parameters chosen by the designer.

5.2 Computation of HDT($\mathbf{e}, t$) of LSPUF

Without loss of generality and for the sake of explanation, we focus only on the LSPUF variant with parameters $s = 0, x = k$ and $m = 1$. This LSPUF configuration is similar to x-XOR APUF, with addition of an input network. In this section, we discuss HDT($\mathbf{e}, t$) property of this LSPUF variant. We consider t flipping bits $\mathbf{c}[i_1], \ldots, \mathbf{c}[i_t]$, where $0 \le i_1 < \cdots < i_t \le n-1$.

Let $Z_{i_1 i_2 \ldots i_t}$ be a random variable which is defined as follows:

$$Z_{i_1 i_2 \ldots i_t} = \begin{cases} 1, & \text{LSPUF output } o \text{ flips when bits } \mathbf{c}[i_1], \ldots, \mathbf{c}[i_t] \text{ flip} \\ 0, & \text{otherwise.} \end{cases}$$

Now we can define HDT($\mathbf{e}, t$) for LSPUF as: $\text{HDT}(\mathbf{e}, t) = \Pr_{\mathbf{c}}(Z_{i_1 i_2 \ldots i_t} = 1) = \Pr_{\mathbf{c}}(\bar{o} \mid \bar{\mathbf{c}}[i_1], \bar{\mathbf{c}}[i_2], \ldots, \bar{\mathbf{c}}[i_t])$, where $\bar{o} = o \oplus 1$ and $\bar{\mathbf{c}}[i_k] = \mathbf{c}[i_k] \oplus 1, 1 \le k \le t$.

Since the input network G generates different inputs $\mathbf{c}_j$ for each APUF A_j, we define a vector $\mathbf{p}_{i_1 \ldots i_t} = (\mathbf{p}_{i_1 \ldots i_t}[0], \ldots, \mathbf{p}_{i_1 \ldots i_t}[x-1])$ to represent the output transition probabilities of all constituent APUFs, where $\mathbf{p}_{i_1 \ldots i_t}[j] = \Pr\left(\bar{r}_j \mid \bar{\mathbf{c}}[i_1], \ldots, \bar{\mathbf{c}}[i_t]\right), j = 0, \ldots, x-1$.

Note that since the input network G and all the flipping positions $i_1, \ldots, i_t$ at $\mathbf{c}$ are known to an adversary, the actual flipping positions at the input vectors of each A_j can be computed in a straightforward way, and then $\mathbf{p}_{i_1 \ldots i_t}[j]$ can be computed accurately as explained in Section 3. Let us define two x-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ where $\mathbf{a}[j] = \mathbf{p}_{i_1 \ldots i_t}[j]$ and $\mathbf{b}[j] = 1 - \mathbf{a}[j], j = 0, \ldots, x-1$. Since the output o of LSPUF flips when y APUF outputs (out of x APUF outputs) flip and y is an odd number, HDT($\mathbf{e}, t$) of a LSPUF can be computed as follows:

Theorem 4. *Let us define a set* $\mathcal{J} = \{j_1, \ldots, j_y\}, 0 \leq j_1 < \cdots < j_y \leq x-1$, *then*

$$\Pr(Z_{i_1 i_2 \ldots i_t} = 1) = \sum_{y=1, y\%2=1}^{x} \sum_{\forall \mathcal{J}} \prod_{j \in \mathcal{J}} \mathbf{a}[j] \prod_{j \notin \mathcal{J}} \mathbf{b}[j]. \tag{16}$$

5.3 Computations of HDT(t) and PC(t) of LSPUF

Here, we also consider the LSPUF variant with parameters $s = 0, x = k$ and $m = 1$. Although, input network was designed to improve mainly HDT($\mathbf{e}$, 1) property of LSPUF, it does not introduce any significant change in HDT(t) and PC(t) properties of an LSPUF. Thus, HDT(t) and PC(t) are similar to XOR APUF, and we mention it briefly in Theorem 5.

Theorem 5. *The* HDT(t) *and* PC(t) *properties of an x-XOR LSPUF with* $x > 2$ *are: (1)* HDT(t) *is approximately* 0.5 *for all* t*, and (2)* PC(t) *is approximately* 0.5 *for all* t.

Proof. The proof of this theorem is similar to the proof of Theorem 3. □

6 SECURITY ANALYSIS OF APUF, XOR APUF AND LSPUF BASED ON HDT($\mathbf{e}, t$) PROPERTY

In Section 2.3, we have mentioned about two adversary threat models: Chosen-challenge Adversary (CCA) and Adaptive Chosen-challenge Adversary (ACCA). Typically, we can develop CCA and ACCA based on the HDT($\mathbf{e}, t$) property. In this paper, we focus on the ACCA based attack because of the following reason: compared to machine learning based CCA, the efficiency (e.g., time and data complexities) of HDT($\mathbf{e}, t$)-based CCA are significantly poor.

Objective of ACCA adversary is to predict the response r_u to an unknown challenge $\mathbf{c}_u$, provided that she can choose another challenge $\mathbf{c}_{\text{chosen}}$ and query the PUF oracle for the corresponding response r. An adversary can achieve her objective based on HDT($\mathbf{e}, t$) property of the target PUF instance, following a two-step scheme as described below:

(1) Firstly, determine a set $\mathcal{E} = \{\mathbf{e}^1, \ldots, \mathbf{e}^d\}$ of patterns $\mathbf{e}$ such that HDT($\mathbf{e}, t$) is poor. The set $\mathcal{E}$ can be constructed from the analysis of HDT($\mathbf{e}, t$) of the target PUF. Let $\mathbf{e}_{\text{best}}$ be the pattern with very poor HDT($\mathbf{e}, t$), and this is the best pattern form the adversary's perspective.
(2) Then, the adversary computes challenge $\mathbf{c}_{\text{chosen}} = \mathbf{c}_u \oplus \mathbf{e}_{\text{best}}$, and then queries the PUF oracle for the response r corresponding to challenge $\mathbf{c}_{\text{chosen}}$. If the value of HDT($\mathbf{e}_{\text{best}}, t$) is close to 1, then $r_u = r \oplus 1$; otherwise, $r_u = r$.

Once the ACCA adversary has the knowledge of CRP ($\mathbf{c}_{\text{chosen}}, r$), she can also derive more unknown CRPs based on pattern $\mathbf{e} \in \mathcal{E} \setminus \mathbf{e}_{\text{best}}$, if HDT($\mathbf{e}, t$) value of $\mathbf{e}$ is significantly poor. The prediction accuracy of ACCA analysis depends on HDT($\mathbf{e}, t$) of the pattern $\mathbf{e}$ that is employed in analysis.

6.1 APUF and XOR APUF under ACCA Threat

From Section 3.2, it can be observed that $\mathbf{e}_0 = (1, 0, \ldots, 0)$ and $\mathbf{e}_{n-1} = (0, \ldots, 0, 1)$ are most useful patterns to the adversary, as they result in significantly very poor HDT($\mathbf{e}$, 1) values. Based on HDT($\mathbf{e}, t$) analysis of APUF in Section 3.3, we can find another interesting pattern $\mathbf{e}_{i,i+1}$ with poor HDT($\mathbf{e}_{i,i+1}$, 2) value, where the pattern vector contains two consecutive '1' bits and all remaining bits are '0', e.g. $\mathbf{e}_{1,2} = (0, 1, 1, 0, \ldots, 0)$. So, an adversary can also employ these patterns to generate related challenge-response pairs using the proposed ACCA attack scheme. In addition, it can be observed that HDT($\mathbf{e}, t$) values for the above mentioned patterns become gradually poor with increasing value of challenge size of the APUF. This implies that accuracy of the ACCA attack is better for larger APUF compared to smaller APUF.

Since the output of an x-XOR APUF is obtained by XOR-ing the outputs of x APUFs, we can also observe poor HDT($\mathbf{e}, t$) property for patterns $\mathbf{e}$ for which APUF shows poor HDT($\mathbf{e}, t$) property.

6.2 LSPUF under ACCA Threat

As the input network in LSPUF can improve HDT($\mathbf{e}_i, 1$) property, we consider HDT($\mathbf{e}_{i,i+1}, 2$) property to develop the ACCA attack on LSPUF. We show that the input network of LSPUF cannot prevent the statistical attack with respect to the pattern $\mathbf{e}_{i,i+1}$. For the sake of explanation, we consider only the input network g_j of the jth constituent APUF A_j. The challenge $\mathbf{c}$ of LSPUF is transformed to $\mathbf{c}_j$ by g_j as described in Section 5.1.1.

As an example, we consider the XOR input network g_0 with intermediate input $\mathbf{d}_0 = \mathbf{c}$ of size 10-bit as follows:

$$\begin{aligned}
&\mathbf{c}_0[0] = \mathbf{d}_0[0] \oplus \mathbf{d}_0[1], && \mathbf{c}_0[1] = \mathbf{d}_0[2] \oplus \mathbf{d}_0[3], && \mathbf{c}_0[2] = \mathbf{d}_0[4] \oplus \mathbf{d}_0[5],\\
&\mathbf{c}_0[3] = \mathbf{d}_0[6] \oplus \mathbf{d}_0[7], && \mathbf{c}_0[4] = \mathbf{d}_0[8] \oplus \mathbf{d}_0[9], && \mathbf{c}_0[5] = \mathbf{d}_0[0],\\
&\mathbf{c}_0[6] = \mathbf{d}_0[1] \oplus \mathbf{d}_0[2], && \mathbf{c}_0[7] = \mathbf{d}_0[3] \oplus \mathbf{d}_0[4], && \mathbf{c}_0[8] = \mathbf{d}_0[5] \oplus \mathbf{d}_0[6],\\
&\mathbf{c}_0[9] = \mathbf{d}_0[7] \oplus \mathbf{d}_0[8].
\end{aligned}$$

Now, we consider pattern $\mathbf{e}_{i,i+1}$ that flips two consecutive bits $(\mathbf{c}[i], \mathbf{c}[i+1])$ as well as $(\mathbf{d}_0[i], \mathbf{d}_0[i+1])$, as $\mathbf{d}_0 = \mathbf{c}$ for input network g_0. It can be observed that two consecutive bits flips in $\mathbf{d}_0$ introduce either two consecutive bit flips in $\mathbf{c}_0$ or only last bit $\mathbf{c}_0[n-1]$ will flip, where $\mathbf{c}_0$ is the input to A_0. This fact is explained with the above example in Table 1.

Table 1. A case of two consecutive bits flips in $\mathbf{d}_0$ and corresponding flips in $\mathbf{c}_0$ of A_0

$(\mathbf{d}_0[0], \mathbf{d}_0[1]) \Rightarrow \{\mathbf{c}_0[5], \mathbf{c}_0[6]\}$,	$(\mathbf{d}_0[1], \mathbf{d}_0[2]) \Rightarrow \{\mathbf{c}_0[0], \mathbf{c}_0[1]\}$,	$(\mathbf{d}_0[2], \mathbf{d}_0[3]) \Rightarrow \{\mathbf{c}_0[6], \mathbf{c}_0[7]\}$,
$(\mathbf{d}_0[3], \mathbf{d}_0[4]) \Rightarrow \{\mathbf{c}_0[1], \mathbf{c}_0[2]\}$,	$(\mathbf{d}_0[4], \mathbf{d}_0[5]) \Rightarrow \{\mathbf{c}_0[7], \mathbf{c}_0[8]\}$,	$(\mathbf{d}_0[5], \mathbf{d}_0[6]) \Rightarrow \{\mathbf{c}_0[2], \mathbf{c}_0[3]\}$,
$(\mathbf{d}_0[6], \mathbf{d}_0[7]) \Rightarrow \{\mathbf{c}_0[8], \mathbf{c}_0[9]\}$,	$(\mathbf{d}_0[7], \mathbf{d}_0[8]) \Rightarrow \{\mathbf{c}_0[3], \mathbf{c}_0[4]\}$,	$(\mathbf{d}_0[8], \mathbf{d}_0[9]) \Rightarrow \{\mathbf{c}_0[9]\}$

In general, it can be observed that if we flip two consecutive bits $(\mathbf{c}[i], \mathbf{c}[i+1])$ in LSPUF input $\mathbf{c}$ with pattern $\mathbf{e}_{i,i+1}$, then there are also flips in bits

$$(\mathbf{d}_j[(i+j)\%n], \mathbf{d}_j[(i+j+1)\%n]),$$

as $\mathbf{d}_j = \mathbf{c} \ggg j$. These bit-flips in $\mathbf{d}_j$ eventually propagate to $\mathbf{c}_j$, which is the input of APUF A_j. The positions of the flipping bits in $\mathbf{c}_j$ depends on flipping bit positions in $\mathbf{d}_j$. Generic version of this observation is stated in the following theorem:

Theorem 6. *If n is an even number and we flip two consecutive bits $(\mathbf{d}_j[u], \mathbf{d}_j[u+1])$, $u = 0, \ldots, n-2$ of the $\mathbf{d}_j$, then there are two consecutive bits flip at $\mathbf{c}_j$, or the last bit $\mathbf{c}_j[n-1]$ in $\mathbf{c}_j$ will be flipped.*

Proof. We consider two different cases:

(1) u **is even:** According to the design of input network, $\mathbf{d}_j[u]$ will flip two bits $\mathbf{c}_j[\frac{u}{2}]$ and $\mathbf{c}_j[\frac{n+u}{2}]$, and $\mathbf{d}_j[u+1]$ will flip two bits $\mathbf{c}_j[\frac{u}{2}]$ and $\mathbf{c}_j[\frac{n+u+2}{2}]$. Thus, when two bits $\mathbf{d}_j[u]$ and $\mathbf{d}_j[u+1]$ flip, it results in flip of two consecutive bits $\mathbf{c}_j[\frac{n+u}{2}]$ and $\mathbf{c}_j[\frac{n+u+2}{2}]$ only.

(2) u **is odd:** According to the design of input network, $\mathbf{d}_j[u]$ will flip two bits $\mathbf{c}_j[\frac{u-1}{2}]$ and $\mathbf{c}_j[\frac{n+u+1}{2}]$, while $\mathbf{d}_j[u+1]$ will flip two bits $\mathbf{c}_j[\frac{u+1}{2}]$ and $\mathbf{c}_j[\frac{n+u+1}{2}]$. Thus, when two bits $\mathbf{d}_j[u]$ and $\mathbf{d}_j[u+1]$ flip, there are only two bits $\mathbf{c}_j[\frac{u-1}{2}]$ and $\mathbf{c}_j[\frac{u+1}{2}]$ which flip. Moreover, these two flipping bits in $\mathbf{c}_j$ are consecutive bits.

Algorithm 2 Prediction of response to $\mathbf{c}$ of (n, k, x)-LSPUF instance P in ACCA

Input: LSPUF instance P
Output: Response o to challenge $\mathbf{c}$
1: $\mathbf{c}_r \leftarrow (\mathbf{c}[0] \oplus 1, \mathbf{c}[1] \oplus 1, \mathbf{c}[2], \ldots, \mathbf{c}[n-1])$ {Related challenge}
2: $o_r = P(\mathbf{c}_r)$ {Evaluate LSPUF with challenge $\mathbf{c}_r$}
3: $o \leftarrow o_r$

A special case occurs when we flip two consecutive bits $(\mathbf{d}_j[n-2], \mathbf{d}_j[n-1])$, and it results in flipping in bits $\mathbf{c}_j[n-1]$ and $\mathbf{c}_j[n]$ (see the case of even u). But bit $\mathbf{c}_j[n]$ does not exist, as n is not a valid index. Thus, only the last bit $\mathbf{c}_j[n-1]$ will flip. □

Thus, for a given challenge $\mathbf{c}$, the adversary can predict the 1-bit response o of LSPUF using Algorithm 2, by flipping two consecutive bits in challenge $\mathbf{c}$, e.g. the first two bits $\mathbf{c}[0]$ and $\mathbf{c}[1]$. It can be observed from experimental result in Section 7.4 that the probability $\Pr(o_r = o)$, where o_r is the response of related challenge $\mathbf{c}_r$, increases with the increasing challenge size n. It implies that the prediction accuracy of the ACCA attack on x-XOR LSPUF is better for longer challenge compared to LSPUF with smaller challenge size.

7 EXPERIMENTAL RESULTS

In this section, we experimentally validate HDT$(\mathbf{e}, t)$, HDT(t) and PC(t) properties of APUF, XOR and LSPUF. We also demonstrate the ACCA attack using HDT$(\mathbf{e}_i, 1)$ and HDT$(\mathbf{e}_{i,i+1}, 2)$ properties.

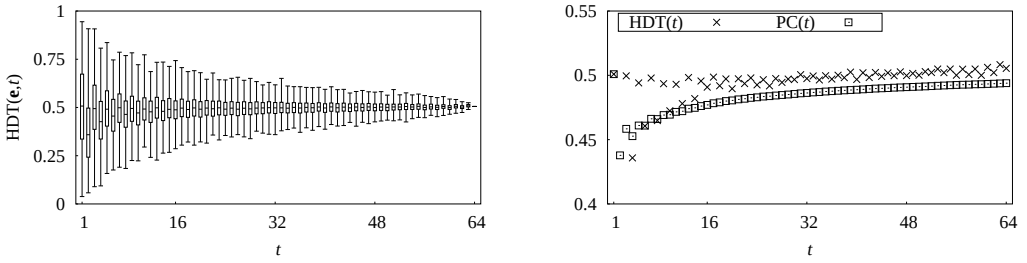
7.1 Experimental Setup

We have performed Matlab simulations and FPGA implementations of above mentioned PUF designs. In case of FPGA implementation, we have used four Xilinx Artix-7 (XC7A100T) FPGAs. Since FPGA based PUFs are not 100% reliable, majority voting (over 11 evaluations of each PUF instance at normal operating condition) is used to generate the golden responses of a PUF instance.

In Matlab simulations, we assume that delay of each delay component follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ [5] with $\mu = 10$ and $\sigma = 0.05$ in $\mathcal{N}(10, \sigma^2)$ to make the all delay values positive. To estimate the effect of noise on simulated APUF behavior (that happens in the real PUF due to the temperature and supply voltage variations), we have employed an additive noise following normal distribution $\mathcal{N}(0, \sigma^2_{\text{noise}})$ [3]. In the presence of noise, each delay component of APUF follows $\mathcal{N}(10, \sigma^2 + \sigma^2_{\text{noise}})$. To control the reliability level of APUF, we have exploited the following relationship between σ and σ_{noise}: $\sigma_{\text{noise}} = \alpha\sigma$, where $0 \leq \alpha \leq 1$, and PUFs are treated to be 100% reliable when $\alpha = 0$.

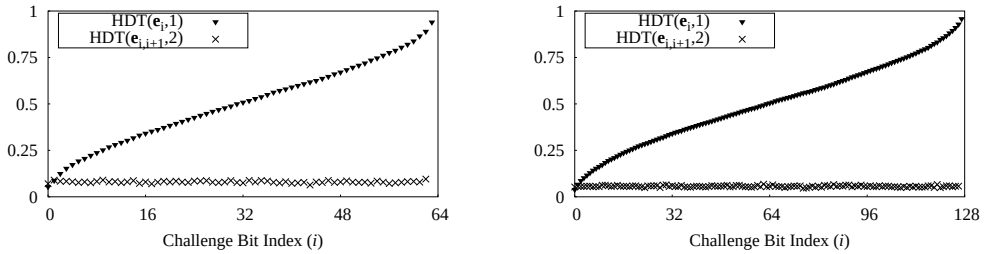
We now describe the computation of HDT$(\mathbf{e}, t)$, which is a fundamental step of HDT(t) and PC(t) computations. For a given pattern $\mathbf{e}_i$, $i \in [0, n-1]$, HDT$(\mathbf{e}_i, 1)$ is the probability of output transition due to flip in $\mathbf{c}[i]$. To estimate this probability, we have evaluated a simulated PUF instance with 5000 random challenge pairs for each value of i, where each challenge pair differs only in the ith bit position (cf. Algorithm 1). We repeated the same experiment on 100 randomly generated simulated PUF instances of a PUF design (e.g. APUF). To make the presentation concise, we provide the average of HDT$(\mathbf{e}_i, 1)$ values for all bit positions i. For FPGA based PUFs, we have used 500 challenge pairs for each challenge bit position to reduce the experiment time. In our discussion, we have used another pattern $\mathbf{e}_{i,i+1}$, and computation of HDT$(\mathbf{e}_{i,i+1}, 2)$ is done in a similar fashion.

The computations of HDT(t) and PC(t) require us to consider all possible patterns $\mathbf{e} = \mathbf{c}_1 \oplus \mathbf{c}_2$, for a given t value, between challenge pair $(\mathbf{c}_1, \mathbf{c}_2)$ with HD$(\mathbf{c}_1, \mathbf{c}_2) = t$. So, these computations are very compute-intensive even for PUF instances with 64-bit challenge, and hence, we consider only



(a) HDT($\mathbf{e}, t$) of 64-bit APUF (b) HDT(t) and PC(t) of 64-bit APUF

Fig. 5. HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties of simulated APUFs.



(a) 64-bit APUF (b) 128-bit APUF

Fig. 6. HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{\mathbf{i,i+1}}, 2$) properties of simulated APUFs. These are computed over 100 APUF instances using 5000 randomly generated CRPs.

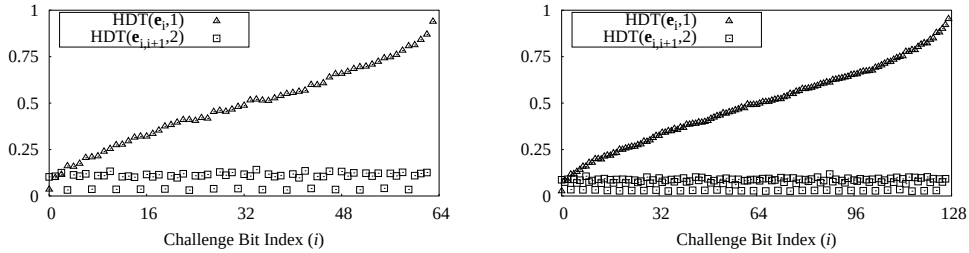
1000 randomly chosen patterns $\mathbf{e}$ for each t value. In computation of HDT(t) for a given t, we need to compute HDT($\mathbf{e}, t$) for each of 1000 randomly chosen $\mathbf{e}$. Finally, PC(t) computation is done based on the computed HDT(t) values.

7.2 Results for APUF

7.2.1 HDT($\mathbf{e}, t$), HDT(t) *and* PC(t) *Properties.* The HDT($\mathbf{e}, t$) property of 64-bit APUF is shown in Fig. 5a based the Matlab simulation. We have used boxplots for each t value, and each boxplot is defined based on 1000 HDT($\mathbf{e}, t$) values corresponding to 1000 random patterns $\mathbf{e}$ with HW($\mathbf{e}$) = t. The whisker of the boxplot spans over the 100% of the dataset. From Fig. 5a, it can be observed that whiskers of the boxplots for $t = 1$ and $t = 2$ extend towards the 0 and 1, and this fact implies that there are a few patterns $\mathbf{e}$ with poor HDT($\mathbf{e}, t$) values. Thus, results of HDT(t) and PC(t) values in Fig. 5b might give wrong impression about robustness against the statistical attacks, as HDT(t) ≈ 0.5 and PC(t) ≈ 0.5 for the most t values.

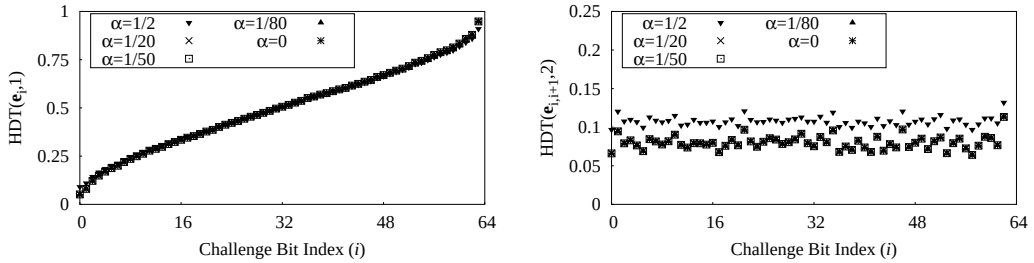
Next, we discuss HDT($\mathbf{e}, t$) property of two specific mismatch patterns $\mathbf{e}_i$ and $\mathbf{e}_{i,i+1}$, as they reveal significant amount of information to an adversary (cf. Section 6.1) .

7.2.2 HDT($\mathbf{e}_i, 1$) *and* HDT($\mathbf{e}_{i,i+1}, 2$) *Properties.* Figures 6 and 7 depict HDT($\mathbf{e}_i, 1$) property of simulated and FPGA implemented APUFs, respectively. It can be observed that HDT($\mathbf{e}_i, 1$) values for very first and last bit positions are significantly poor, as pointed out in Section 3.2. The adversary can



(a) APUF 64-bit (b) APUF 128-bit

Fig. 7. HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{i,i+1}, 2$) properties of FPGA implemented APUF.



(a) HDT($\mathbf{e}_i, 1$) with noise (b) HDT($\mathbf{e}_{i,i+1}, 2$) with noise

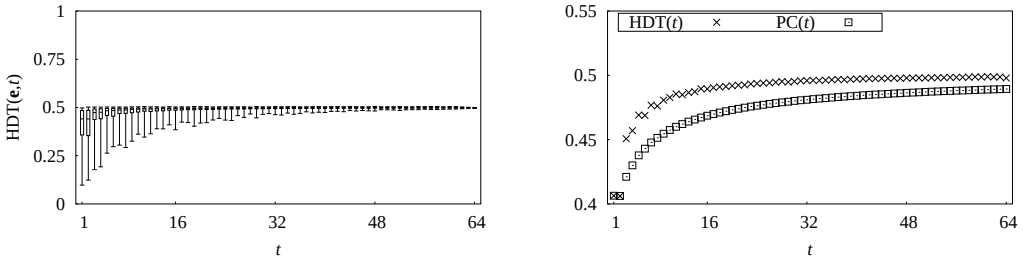
Fig. 8. Delineating the robustness of HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{i,i+1}, 2$) for 64-bit APUFs in the presence of noise following $\mathcal{N}(0, \sigma_{\text{noise}})$ and $\sigma_{\text{noise}} = \alpha\sigma$ where σ is the standard deviation of delays of APUF's delay components.

Table 2. Reliability (%) of simulated 64-bit APUF, x-XOR APUF and x-XOR LSPUF with $\sigma_{\text{noise}} = \alpha\sigma$

PUF	x	Reliability (Avg.)				
		$\alpha = 1/2$	$\alpha = 1/20$	$\alpha = 1/50$	$\alpha = 1/80$	$\alpha = 0$
APUF	–	94.80	99.47	99.79	99.86	100
XOR APUF	2	87.87	98.93	99.58	99.73	100
	3	80.83	98.38	99.36	99.59	100
LSPUF	2	87.89	98.90	99.58	99.73	100
	3	80.80	98.36	99.36	99.61	100

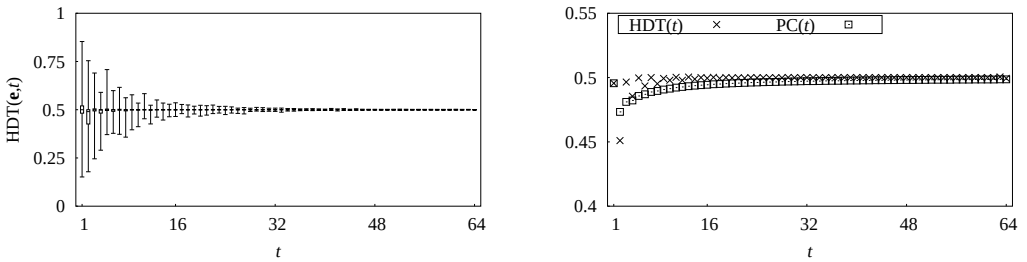
exploit these observations to derive new CRPs from the previously revealed CRPs (cf. Section 6.1). This will reduce the size of effective secure CRP space.

In [11], it was reported that HDT($\mathbf{e}_i, 1$) property of APUF can be improved using their a XOR based input network. Although this is true, it does not improve HDT($\mathbf{e}, t$) property of APUF for arbitrary values of t. Figures 6 and 7 also show the relatively poor HDT($\mathbf{e}_{i,i+1}, 2$) property for APUF. This observation is in agreement with the theoretical conclusion reached in Section 3.3. Later in Fig. 17, we show that the input network in [11] cannot achieve a good HDT($\mathbf{e}_{i,i+1}, 2$) for APUF. Another important observation from the experimental results is that both the values of HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{i,i+1}, 2$) of APUFs become poorer with increasing value of challenge size n.



(a) HDT(**e**, t) of 2-XOR APUF

(b) HDT(t) and PC(t) of 2-XOR APUF



(c) HDT(**e**, t) of 3-XOR APUF

(d) HDT(t) and PC(t) of 3-XOR APUF

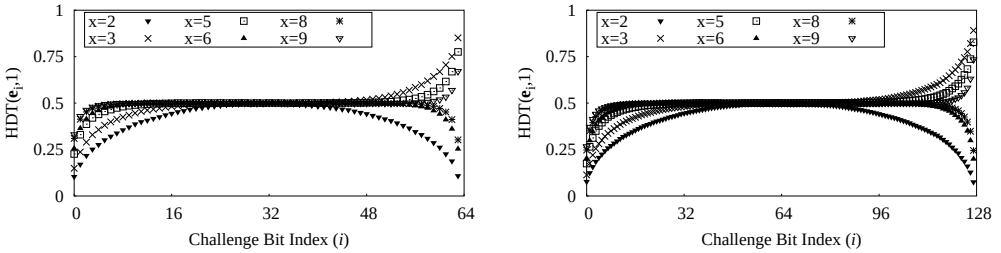
Fig. 9. HDT(**e**, t), HDT(t) and PC(t) properties of simulated 64-bit x-XOR APUF for $x = 2, 3$. For even values of x, HDT(**e**, t) follows the trend as in (a), and for odd values of x, HDT(**e**, t) follows the trend as in (c).

Since HDT(**e**, t) computation relies on CRPs of a PUF instance, the reliability of PUF has an influence on HDT(**e**, t) property. To observe this fact for HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2), we have considered 64-bit simulated APUF with additive noise (cf. Section 7.1). Reliability values of APUF for different σ_{noise} values are reported in Table 2. The HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) values of 64-bit simulated APUFs are depicted in Fig. 8 for different α values. For $\alpha = 1/2$, reliability of APUF is approximately 94%, and as a consequence we can see a small shift in HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) values. It is evident that HDT(**e**, t) value is affected by the reliability, but effect is not significant for an APUF unless reliability is very poor.

7.3 Results for XOR APUF

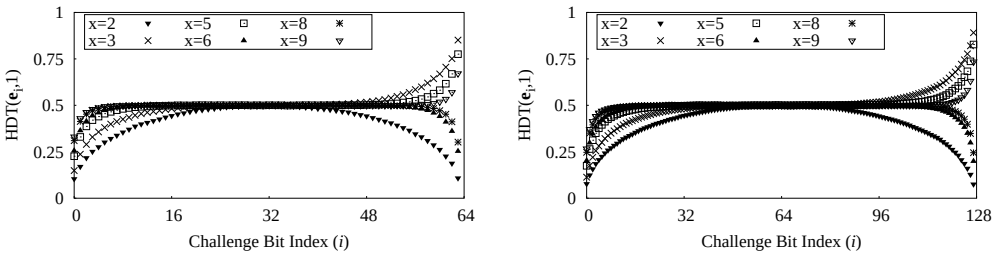
Like the case of APUF, for XOR APUF, we have performed a similar experiment to compute HDT(t) and PC(t) properties for simulated 64-bit 2-XOR APUF and 3-XOR APUF, and the corresponding results are reported in Figs. 9b and 9d, respectively. Detailed results for HDT(**e**, t) based on 1000 random patterns **e** for each t value are reported in Figs. 9a and 9c using boxplots. From Figs. 9a and 9c, it is evident that there are a few patterns **e** with $t = 1, 2$ that are useful to an adversary. We have reported HDT($\mathbf{e}_i$, 1) properties of simulated XOR APUFs (64-bit and 128-bit) in Fig. 11, which is also known as SAC property.

The result of HDT(t) property for XOR APUF had been reported in [11], but authors considered only the HD of challenge pairs regardless of their points of mismatch (i.e., pattern **e**). An adversary can try to find the particular mismatch pattern of challenge pairs, from which she can get comparatively more statistical information to reduce the unpredictability property of XOR APUF.



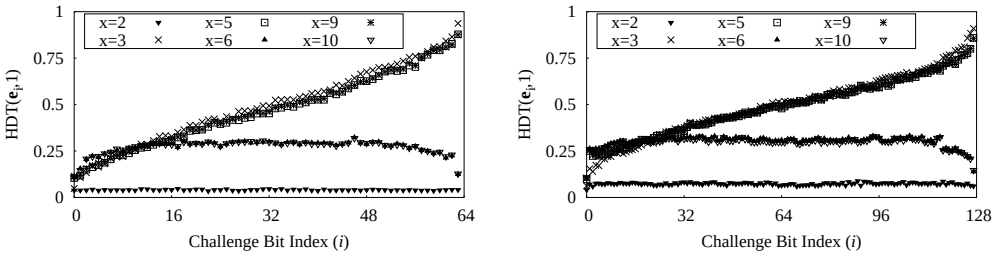
(a) 64-bit x-XOR APUF (b) 128-bit x-XOR APUF

Fig. 10. HDT($\mathbf{e}_i$, 1) property of simulated x-XOR APUF.



(a) 64-bit x-XOR APUF (b) 128-bit x-XOR APUF

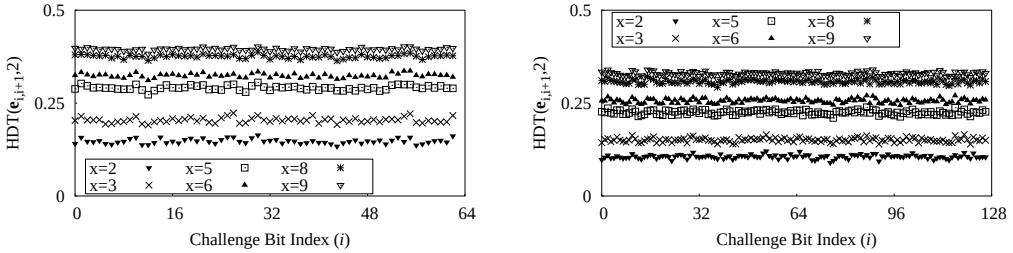
Fig. 11. HDT($\mathbf{e}_i$, 1) property of simulated x-XOR APUF.



(a) 64-bit x-XOR APUF (b) 128-bit x-XOR APUF

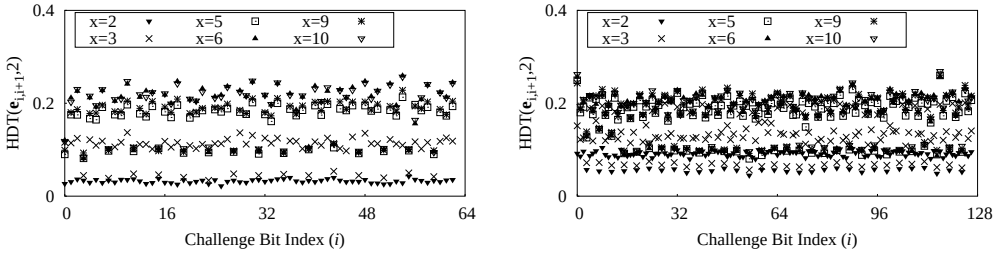
Fig. 12. HDT($\mathbf{e}_i$, 1) property of FPGA implemented x-XOR APUF.

The HDT($\mathbf{e}_i$, 1) results for FPGA implemented XOR APUF are reported in Fig. 12. To the best of our knowledge, there has been no previous published result for FPGA-based XOR APUF in the context of HDT($\mathbf{e}_i$, 1) property. We observed that HDT($\mathbf{e}_i$, 1) of simulated x-XOR APUF (cf. Fig. 11) is different from that of the FPGA-implemented x-XOR APUF (cf. Fig. 12) for odd values of x, as FPGA-based APUF has very poor uniqueness. In particular, APUF instances on a single board have almost similar challenge-response behavior [9, Table 6]. Hence, when x is odd, the FPGA-based



(a) 64-bit x-XOR APUF (b) 128-bit x-XOR APUF

Fig. 13. HDT($\mathbf{e}_{i,i+1}$, 2) property of simulated x-XOR APUF.



(a) 64-bit x-XOR APUF (b) 128-bit x-XOR APUF

Fig. 14. HDT($\mathbf{e}_{i,i+1}$, 2) property of FPGA implemented x-XOR APUF.

x-XOR APUF behaves similar to a single APUF. This is the reason why the plot of HDT($\mathbf{e}_i$, 1) of x-XOR APUF with odd x is similar to that for APUF.

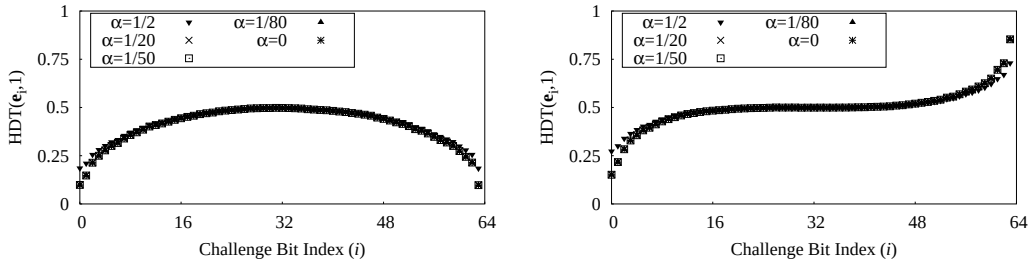
We have also performed HDT($\mathbf{e}_{i,i+1}$, 2) property of both simulated and FPGA implemented XOR APUF, and the corresponding results are reported in Figs. 13 and 14, respectively. Results shows that HDT($\mathbf{e}_{i,i+1}$, 2) property is also poor compared to its ideal value 0.5.

It is worth mentioning that both HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) properties of x-XOR APUF with n-bit challenge improve with larger values of x, whereas larger n can be an influencing factor to reducing HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) properties. An adversary can exploit both HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) properties of XOR APUF to perform a statistical attack—the ACCA attack.

Like APUF, HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) values of XOR APUF are affected by the reliability. To demonstrate this fact, we simulated 64-bit 2-XOR APUF and 3-XOR APUF with (additive) noise in consideration, and corresponding HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) values are shown in Fig. 15. Reliability values of 2-XOR APUF and 3-XOR APUF for different σ_{noise} values are mentioned in Table 2. The changes in HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) values are not significant within the typical acceptable reliability range.

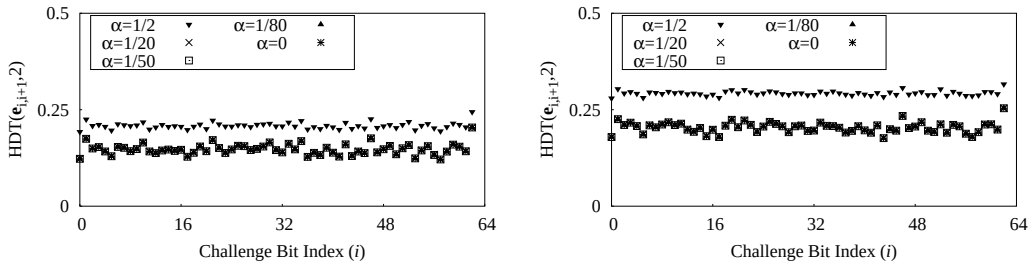
7.4 Results for LSPUF

Like APUF and XOR APUF, we also performed a detailed experiment to investigate HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties for simulated 64-bit 2-XOR LSPUF and 3-XOR LSPUF, and the corresponding results are reported in Fig. 16. Boxplots in Figs. 16a and 16c have a different trend



(a) HDT($\mathbf{e}_i$, 1) of 2-XOR APUF

(b) HDT($\mathbf{e}_i$, 1) of 3-XOR APUF



(c) HDT($\mathbf{e}_{i,i+1}$, 2) of 2-XOR APUF

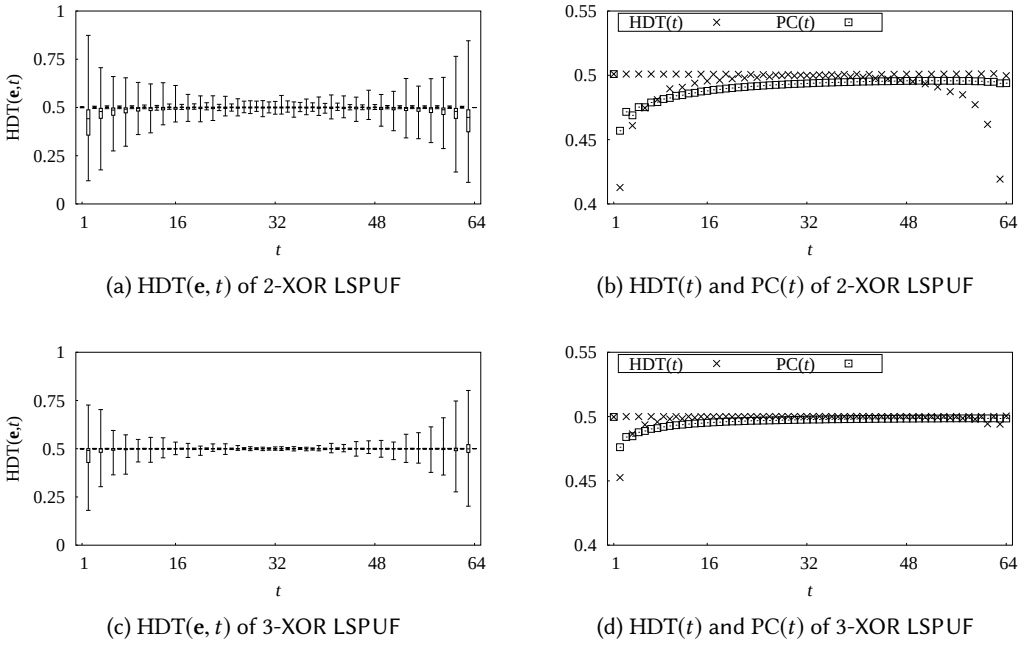
(d) HDT($\mathbf{e}_{i,i+1}$, 2) of 3-XOR APUF

Fig. 15. Delineating the robustness of HDT($\mathbf{e}_i$, 1) and HDT($\mathbf{e}_{i,i+1}$, 2) for simulated 64-bit XOR APUFs in the presence of noise, and $\sigma_{\text{noise}} = \alpha\sigma$.

than that of APUF (cf. Fig. 5a) and XOR APUF (cf. Figs. 9a and 9c) due to the input network of LSPUF. From Figs. 16a and 16c, it can be observed that HDT($\mathbf{e}, t$) values are approximately 0.5 for odd $t \in [1, 31]$ and even $t \in [34, 64]$. In other cases of t, HDT($\mathbf{e}, t$) are gradually improving when $t \rightarrow n/2$. Effects of poor HDT($\mathbf{e}, t$) for some t values can be seen in corresponding HDT(t) and PC(t) results as depicted in Figs. 16b and 16d. Next, we discuss HDT($\mathbf{e}, t$) for a specific mismatch pattern $\mathbf{e}_{i,i+1}$ that reveal significant information to an adversary.

The input network in LSPUF design was introduced to achieve good HDT($\mathbf{e}_i$, 1) property for LSPUF outputs, by improving HDT($\mathbf{e}_i$, 1) property (also known as SAC) for used APUFs, i.e., HDT($\mathbf{e}_i$, 1) of APUF instances in presence of the input network is around 0.5 [11]. From the results in Figs. 16a and 16c, one can observe the following fact: the input network of LSPUF in [11] can ensure good HDT($\mathbf{e}, t$) for 50% of t values (cf. Figs. 16a and 16c), but it cannot improve HDT($\mathbf{e}, t$) property in general. This is a weakness of the LSPUF input network. In this work, we have shown that HDT($\mathbf{e}_{i,i+1}$, 2) properties of APUF (cf. Sections 3.3 and 7.2) and LSPUF (cf. Section 5) are poor.

Now, we discuss results regarding HDT($\mathbf{e}_{i,i+1}$, 2) property of the LSPUF. Without loss of generality, we have computed HDT($\mathbf{e}_{i,i+1}$, 2) of the first APUF (A_0) of FPGA implemented LSPUF with input network $\mathbf{d}_0 = g_0(\mathbf{c} \ggg 0)$. This result is reported in Fig. 17. It is evident that the HDT($\mathbf{e}_{i,i+1}$, 2) property is comparatively poor (i.e. approximately 0.1 for 64-bit and 0.07 for 128-bit APUFs) than its ideal value 0.5. In this case, HDT($\mathbf{e}_{i,i+1}$, 2) property of APUF are similar for every consecutive bit pair, excluding the most significant bit pair. As discussed in Theorem 6, when there is a flip in the last bit pair of challenge $\mathbf{c}$, it flips the most significant bit of input $\mathbf{c}_0$ to A_0 (cf. Fig. 4). So, HDT($\mathbf{e}_{i,i+1}$, 2) of this last pair is equivalent to HDT($\mathbf{e}_i$, 1) of last bit for APUF without input network.



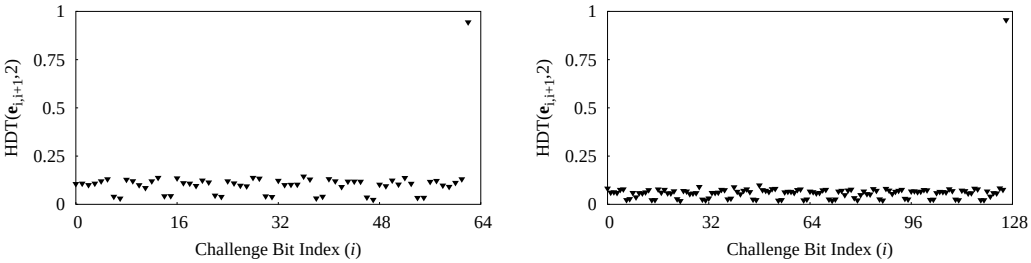
(a) HDT(**e**, t) of 2-XOR LSPUF

(b) HDT(t) and PC(t) of 2-XOR LSPUF

(c) HDT(**e**, t) of 3-XOR LSPUF

(d) HDT(t) and PC(t) of 3-XOR LSPUF

Fig. 16. HDT(**e**, t), HDT(t) and PC(t) properties of simulated 64-bit x-XOR LSPUF for $x = 2, 3$.



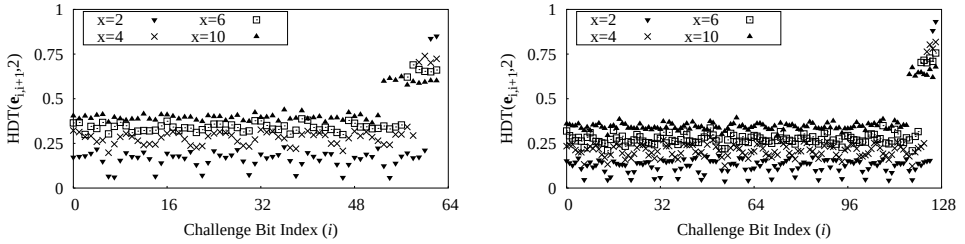
(a) 64-bit APUF

(b) 128-bit APUF

Fig. 17. HDT($\mathbf{e}_{i,i+1}$, 2) properties of FPGA implemented APUF with LSPUF's input network.

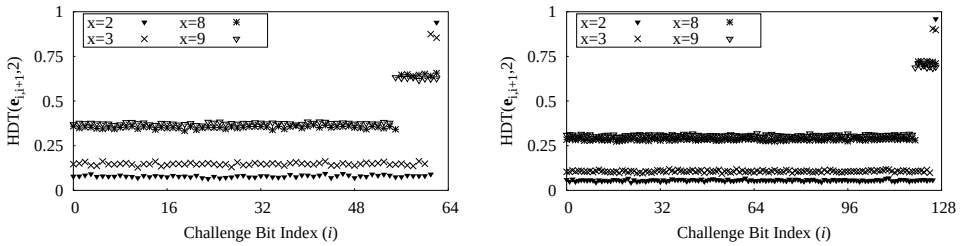
This implies that the input network of LSPUF needs to be modified to achieve the ideal value of HDT(**e**, t) = 0.5. Otherwise, it becomes a threat to LSPUF security (cf. Section 6).

As a consequence of the poor HDT($\mathbf{e}_{i,i+1}$, 2) property of constituent APUFs, LSPUF also exhibits poor HDT($\mathbf{e}_{i,i+1}$, 2) property. Figure 18 depicts HDT($\mathbf{e}_{i,i+1}$, 2) property of FPGA implemented LSPUF (64-bit and 128-bit) with 1-bit output ($m = 1$). The output bit is generated similar to an x-XOR APUF, for different values $x = 2, \ldots, 9$. We have also performed the same experiment using Matlab based simulation of LSPUFs to observe the platform independent behavior, and results are reported in Fig. 19. The following two facts can be observed from the experimental results:



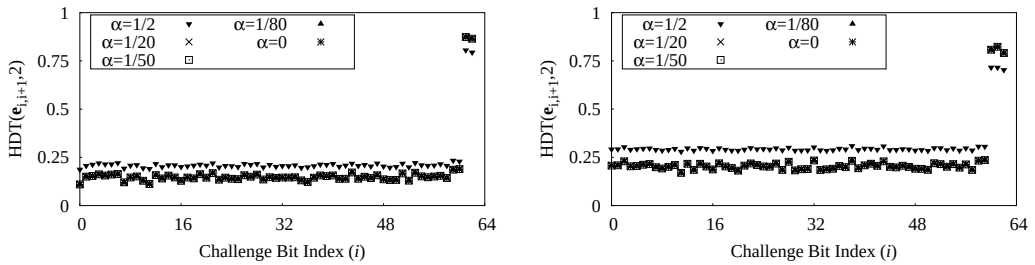
(a) 64-bit x-XOR LSPUF (b) 128-bit x-XOR LSPUF

Fig. 18. HDT($\mathbf{e}_{i,i+1}, 2$) property of FPGA implemented LSPUF with one output bit. This case is similar as XOR APUF with input network.



(a) 64-bit x-XOR LSPUF (b) 128-bit x-XOR LSPUF

Fig. 19. HDT($\mathbf{e}_{i,i+1}, 2$) property of simulated LSPUF with 1-bit output ($m = 1$). This case is similar as XOR APUF with input network.



(a) HDT($\mathbf{e}_{i,i+1}, 2$) of 2-XOR LSPUF (b) HDT($\mathbf{e}_{i,i+1}, 2$) of 3-XOR LSPUF

Fig. 20. Delineating the robustness of HDT($\mathbf{e}_{i,i+1}, 2$) for 64-bit LSPUFs in the presence of noise, and $\sigma_{\text{noise}} = \alpha\sigma$.

(1) The HDT($\mathbf{e}_{i,i+1}, 2$) properties of APUF and LSPUF become poorer with the increasing value of challenge length n. This implies that the leakage of information is more for longer PUFs upon revealing a set of CRPs.
(2) Improvement in HDT($\mathbf{e}_{i,i+1}, 2$) property of LSPUF can be observed with increasing value of x. We can see that the HDT($\mathbf{e}_{i,i+1}, 2$) property of LSPUF for $x = 9$ (cf. Fig. 19) is 10-20%

Table 3. A list of useful pattern vectors $\mathbf{e}$ w.r.t. HDT($\mathbf{e}, t$) property for APUF

$n^{\dagger}$	Output Transition Probability [HDT($\mathbf{e}, t$)]							
	Simulation				FPGA			
	$\mathbf{e}_0$	$\mathbf{e}_{n-1}$	$\mathbf{e}_{0,1}$	$\mathbf{e}_{n-2,n-1}$	$\mathbf{e}_0$	$\mathbf{e}_{n-1}$	$\mathbf{e}_{0,1}$	$\mathbf{e}_{n-2,n-1}$
64	0.054	0.940	0.070	0.095	0.035	0.939	0.103	0.126
128	0.039	0.958	0.053	0.057	0.026	0.953	0.087	0.093
256	0.026	0.972	0.036	0.042	⋆	⋆	⋆	⋆
512	0.020	0.981	0.027	0.027	⋆	⋆	⋆	⋆

† Challenge size

⋆ It implies that corresponding data in not available, as we did not implement 256-bit and 512-bit APUFs on Xilinx FPGA.

apart from its ideal value 50%. However, in practice we cannot use larger x values due to reliability issues [2, 5].

To observe the effect of reliability on HDT($\mathbf{e}_{i,i+1}, 2$) for LSPUF, we have simulated 64-bit 2-XOR LSPUF and 3-XOR LSPUF with additive noise, as discussed in Section 7.1, and the reliability of LSPUF is reported in Table 2. Fig. 20 shows HDT($\mathbf{e}_{i,i+1}, 2$) of LSPUF with different values for σ_{noise}, and a shift in HDT($\mathbf{e}_{i,i+1}, 2$) can be observed when reliability of LSPUF is significantly poor. Thus, HDT($\mathbf{e}_{i,i+1}, 2$) values are robust within the acceptable reliability range.

7.5 Demonstration of Proposed ACCA Attack

As we mentioned earlier in Section 6 that objective of an ACCA adversary is to predict the response to a challenge using the response obtained by applying another chosen challenge. Table 3 lists a few useful pattern vectors $\mathbf{e}$ with HDT($\mathbf{e}, t$) value for APUF that can be used by the adversary for choosing a challenge for the ACCA attack. There might be many such useful pattern vectors $\mathbf{e}$. Here we demonstrate this fact with experimental results.

Now we describe the ACCA attack. Let $\mathbf{c}$ be an unknown challenge and the adversary predicts the response to this challenge. She can choose a challenge $\mathbf{c}_{\text{chosen}} = \mathbf{c} \oplus \mathbf{e}_0$ and queries the PUF oracle for its response, denoted by r_{chosen}. Thus, the response to the challenge $\mathbf{c}$ is $r = r_{\text{chosen}}$, as for $\mathbf{e}_0$ output transition probability (HDT($\mathbf{e}_0, t$)) approaches 0 (cf. Table 3). If she would choose pattern $\mathbf{e}_{n-1}$ (i.e. $\mathbf{c}_{\text{chosen}} = \mathbf{c} \oplus \mathbf{e}_{n-1}$) instead of $\mathbf{e}_0$, then response to $\mathbf{c}$ would be $r = r_{\text{chosen}} \oplus 1$, as HDT($\mathbf{e}_{n-1}, t$) approaches 1 (cf. Table 3).

The prediction accuracy of the ACCA attack is reported in Table 4 for simulated APUF designs. In this simulation, we have used 50,000 CRPs and derived another 50,000 CRPs for each pattern $\mathbf{e}$ listed in Table 3 by following the ACCA attack scheme. We have also shown the influence of reliability on the prediction accuracy in Table 4 using simulated additive noise (cf. Section 7.1). It can be observed from Table 4 that prediction accuracy improves with the increasing challenge size of PUF, i.e., prediction accuracy of 512-bit APUF is better than that of 64-bit APUF. From the observed results, it is clear that the attack experimental results corroborate the proposed ACCA attack methodology.

Since the CCA attack discussed in Section 6 is not more efficient than ML-based modeling attacks, we would exclude the experimental validation of CCA attack in this paper, but we discuss how an adversary can generate related CRPs from a given CRP. Let $(\mathbf{c}, r)$ be a known CRP of a n-bit APUF and then adversary can derive following CRPs using the pattern vectors reported in Table 3: $(\mathbf{c} \oplus \mathbf{e}_0, r)$, $(\mathbf{c} \oplus \mathbf{e}_{n-1}, r \oplus 1)$, $(\mathbf{c} \oplus \mathbf{e}_{0,1}, r)$ and $(\mathbf{c} \oplus \mathbf{e}_{n-2,n-1}, r)$. Thus, the effective CRP space of an APUF can be reduced significantly after revealing some set of CRPs to the adversary. This is a threat to APUF based authentication protocols where the adversary can eavesdrop CRPs of an APUF instance.

Table 4. Prediction accuracy of the ACCA attack on simulated APUFs

$n^\star$	$\alpha^\dagger$	Prediction Accuracy (Avg.,Std.) [%]				Reliability (Avg.,Std.) [%]
		$\mathbf{e}_0$	$\mathbf{e}_{n-1}$	$\mathbf{e}_{0,1}$	$\mathbf{e}_{n-2,n-1}$	
64	1/2	(90.31,3.59)	(90.91,2.66)	(88.79,5.28)	(89.74,3.04)	(94.97,0.48)
	1/20	(93.41,5.11)	(94.20,4.30)	(91.55,7.19)	(92.34,4.64)	(99.49,0.06)
	1/50	(93.48,5.17)	(94.24,4.34)	(91.58,7.23)	(92.36,4.67)	(99.80,0.03)
	1/80	(93.48,5.17)	(94.24,4.34)	(91.58,7.25)	(92.35,4.67)	(99.87,0.02)
	0	(93.48,5.17)	(94.24,4.35)	(91.59,7.26)	(92.35,4.67)	(100,0)
128	1/2	(91.87,1.26)	(91.95,1.53)	(91.61,1.86)	(90.44,2.72)	(95.05,0.31)
	1/20	(95.89,2.84)	(96.00,3.05)	(95.40,3.36)	(93.49,4.40)	(99.50,0.04)
	1/50	(95.94,2.89)	(96.06,3.12)	(95.45,3.43)	(93.52,4.43)	(99.80,0.02)
	1/80	(95.93,2.88)	(96.07,3.11)	(95.45,3.44)	(93.51,4.43)	(99.87,0.02)
	0	(95.94,2.89)	(96.08,3.12)	(95.46,3.45)	(93.51,4.43)	(100,0)
256	1/2	(92.33,0.71)	(92.16,0.94)	(91.93,1.38)	(92.04,1.28)	(94.92,0.26)
	1/20	(96.98,1.90)	(96.55,2.30)	(96.18,2.74)	(96.49,2.66)	(99.49,0.05)
	1/50	(97.03,1.96)	(96.61,2.38)	(96.23,2.78)	(96.56,2.71)	(99.79,0.02)
	1/80	(97.04,1.95)	(96.63,2.38)	(96.23,2.77)	(96.56,2.71)	(99.87,0.02)
	0	(97.04,1.96)	(96.63,2.40)	(96.24,2.78)	(96.57,2.72)	(100,0)
512	1/2	(92.63,0.37)	(92.68,0.46)	(92.46,1.03)	(92.47,0.81)	(94.98,0.20)
	1/20	(97.82,1.20)	(98.09,1.37)	(97.41,2.26)	(97.44,2.07)	(99.49,0.04)
	1/50	(97.92,1.31)	(98.23,1.49)	(97.51,2.33)	(97.55,2.16)	(99.79,0.02)
	1/80	(97.92,1.32)	(98.26,1.51)	(97.52,2.34)	(97.57,2.17)	(99.87,0.02)
	0	(97.93,1.33)	(98.27,1.53)	(97.52,2.34)	(97.58,2.19)	(100,0)

* Challenge size

† $\sigma_{\text{noise}} = \alpha\sigma$

Note: This result is obtained based on 50 different simulated instances of a APUF design with n-bit challenge. For each PUF instance, we considered 50,000 CRPs, and derived another 50,000 CRPs with prediction accuracy reported above by following the ACCA modeling.

Like in Table 3, one can list useful patterns with its HDT($\mathbf{e}, t$) for XOR APUF based on the results reported in Section 7.3. In case of LSPUF, a useful pattern to an ACCA adversary is the $\mathbf{e}_{i,i+1}$ (cf. Section 7.4), as HDT($\mathbf{e}_i, 1$) is improved by using the input network. However, PUF designer can prevent this kind of statistical attack using large value for x in case of x-XOR APUF and x-XOR LSPUF, as HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{i,i+1}, 2$) approach 0.5. Whereas in case of large value of n (challenge length), attack is more efficient, as HDT($\mathbf{e}_i, 1$) and HDT($\mathbf{e}_{i,i+1}, 2$) become poor.

8 CONCLUSIONS

In this work, we have proposed a new test, which we term as HDT($\mathbf{e}, t$) test, for evaluating the unpredictability property of PUFs. As case studies, we have provided a comprehensive study of HDT($\mathbf{e}, t$), HDT(t) and PC(t) properties of APUF, XOR APUF, and LSPUF. We have also shown that HDT($\mathbf{e}, t$) test is more general in comparison with HDT(t) and PC(t) test schemes—if a PUF design qualifies the HDT($\mathbf{e}, t$) property, then it also satisfies the HDT(t) and PC(t), but the reverse does not hold. To validate HDT($\mathbf{e}, t$) properties of APUF, XOR APUF and LSPUF, we have simulated and implemented the designs on Xilinx FPGA platform. Our simulation and implementation results are in agreement with our theoretical observations. We have also developed an ACCA attack where adversary can predict the response to an unknown challenge based on the response to a chosen challenge, and we have demonstrated the ACCA attack on simulated and FPGA-implemented APUFs. This attack reduces the unpredictability properties of APUF, XOR APUF and LSPUF designs. Our future work would be directed at utilizing the concepts developed in this paper to provide a

testing and evaluation methodology for the unpredictability property of PUF design in the context of statistical attacks.

APPENDIX

A ANALYTICAL EXPRESSION OF HDT($\mathbf{e}, t$) FOR APUF

We now proceed to establish analytical expression for HDT($\mathbf{e}, t$) with respect to t flipping bits $\mathbf{c}[i_1], \ldots, \mathbf{c}[i_t]$ flip. Let us define a partition of $n+1$ indices (cf. Fig. 21) in $\mathbf{\Phi}$ into $t+1$ sets $\mathcal{I}_1, \ldots, \mathcal{I}_{t+1}$, where $\mathcal{I}_1 = \{0, \ldots, i_1\}$ and $|\mathcal{I}_1| = i_1 + 1$; $\mathcal{I}_k = \{i_{k-1} + 1, \ldots, i_k\}$ and $|\mathcal{I}_k| = i_k - i_{k-1}, k = 2, \ldots, t$; $\mathcal{I}_{t+1} = \{i_t + 1, \ldots, n\}$ and $|\mathcal{I}_{t+1}| = n - i_t$. Now we consider the following cases based on the parity of t as follows.

Case-I: t is even. Without loss of generality and for the sake of explanation, we focus on the case $t = 2$, i.e., the case where only two challenge bits $\mathbf{c}[i_1]$ and $\mathbf{c}[i_2]$ flip. Then, according to Fig. 21, there are three sets $\mathcal{I}_1, \mathcal{I}_2$ and $\mathcal{I}_3$. Since all the $\mathbf{\Phi}[j], j \in \mathcal{I}_1 \cup \mathcal{I}_3$ have an even number of flipping bits, the signs of these $\mathbf{\Phi}[j]$ terms are not changed when $\mathbf{c}[i_1]$ and $\mathbf{c}[i_2]$ flip (cf. Observation 1). It implies that $\mathcal{B}_{i_1 i_2} = \mathcal{I}_1 \cup \mathcal{I}_3$. All the $\mathbf{\Phi}[j], j \in \mathcal{I}_2$ have an odd number of flipping bits, and the signs of these $\mathbf{\Phi}[j]$ are changed when $\mathbf{c}[i_1]$ and $\mathbf{c}[i_2]$ flip. Hence, $\mathcal{A}_{i_1 i_2} = \mathcal{I}_2$. Figure 21a describes the non-flipping zones ($\mathcal{N}$) and the flipping zones ($\mathcal{F}$). We generalize this fact in Theorem 7:

THEOREM 7. *If there are t flipping bits $\mathbf{c}[i_1], \ldots, \mathbf{c}[i_t]$ and t is even, then $\mathcal{A}_{i_1 i_2 \ldots i_t} = \mathcal{I}_2 \cup \mathcal{I}_4 \cup \cdots \cup \mathcal{I}_{t-2} \cup \mathcal{I}_t$ and $\mathcal{B}_{i_1 i_2 \ldots i_t} = \mathcal{I}_1 \cup \mathcal{I}_3 \cup \cdots \cup \mathcal{I}_{t-1} \cup \mathcal{I}_{t+1}$, and $|\mathcal{A}_{i_1 i_2 \ldots i_t}| = (i_2 - i_1) + (i_4 - i_3) + \cdots + (i_{t-2} - i_{t-3}) + (i_t - i_{t-1}) + i_{t-2} - i_{t-1} + i_t$ and $|\mathcal{B}_{i_1 i_2 \ldots i_t}| = (i_1 + 1) + (i_3 - i_2) + (i_5 - i_4) + \cdots + (i_{t-1} - i_{t-2}) + (n - i_t)$.*

Case-II: t is odd. Again without loss of generality and for the sake of explanation, we focus on $t = 3$, i.e., only three bits $\mathbf{c}[i_1]$, $\mathbf{c}[i_2]$ and $\mathbf{c}[i_3]$ flip. With the same argument, we have $\mathcal{A}_{i_1 i_2 i_3} = \mathcal{I}_1 \cup \mathcal{I}_3$ and $\mathcal{B}_{i_1 i_2 i_3} = \mathcal{I}_2 \cup \mathcal{I}_4$ as depicted in Theorem 8.

We generalize this fact in Theorem 8:

THEOREM 8. *If there are t flipping bits $\mathbf{c}[i_1], \ldots, \mathbf{c}[i_t]$ and t is odd, then $\mathcal{A}_{i_1 i_2 \ldots i_t} = \mathcal{I}_1 \cup \mathcal{I}_3 \cup \cdots \cup \mathcal{I}_{t-2} \cup \mathcal{I}_t$ and $\mathcal{B}_{i_1 i_2 \ldots i_t} = \mathcal{I}_2 \cup \mathcal{I}_4 \cup \cdots \cup \mathcal{I}_{t-1} \cup \mathcal{I}_{t+1}$, and $|\mathcal{A}_{i_1 i_2 \ldots i_t}| = (i_1 + 1) + (i_3 - i_2) + \cdots + (i_{t-2} - i_{t-3}) + (i_t - i_{t-1})$ and $|\mathcal{B}_{i_1 i_2 \ldots i_t}| = (i_2 - i_1) + (i_4 - i_3) + (i_6 - i_5) + \cdots + (i_{t-1} - i_{t-2}) + (n - i_t)$.*

From Eq. (10), we can write HDT($\mathbf{e}_{i_1, \ldots, i_t}, t$) = $\Pr(X_{i_1 i_2 \ldots i_t} = 1) = \Pr(|\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}}| > |\Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}}|)$. Assume that $|\mathcal{A}_{i_1 i_2 \ldots i_t}| = h$, then $|\mathcal{B}_{i_1 i_2 \ldots i_t}| = n - h + 1$. In other words, $\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} \sim \mathcal{N}(0, h\sigma^2)$ and $\Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}} \sim \mathcal{N}(0, (n - h + 1)\sigma^2)$.

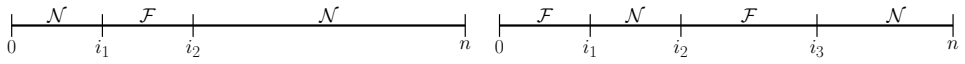
Since Gaussian function is symmetric with respect to mean (μ), and $\mu = 0$ for both $\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}}$ and $\Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}}$, the probability $\Pr(X_{i_1 i_2 \ldots i_t} = 1) = 4p$, where $p = \Pr(\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} > \Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}} | \Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}} \geq 0, \Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}} \geq 0)$. According to the probability theory, $p = \int_0^\infty \phi_{0,(n-h+1)\sigma^2}(u) \Phi_{0,h\sigma^2}(-u)\, \mathrm{d}u$, and $\Pr(X_{i_1 i_2 \ldots i_t} = 1)$ can be expressed as:

$$\Pr(X_{i_1 i_2 \ldots i_t} = 1) = \Pr(|\Delta_{\mathcal{A}_{i_1 i_2 \ldots i_t}}| > |\Delta_{\mathcal{B}_{i_1 i_2 \ldots i_t}}|) = 4 \times \int_0^\infty \phi_{0,(n-h+1)\sigma^2}(u) \Phi_{0,h\sigma^2}(-u)\, \mathrm{d}u.$$

B PROOF OF THEOREM 1

PROOF. For the sake of explanation, let us denote the flipping and non-flipping zones by $\mathcal{F}_{\mathbf{e}} (= \mathcal{A}_{i_1 \ldots i_t})$ and $\mathcal{N}_{\mathbf{e}} (= \mathcal{B}_{i_1 \ldots i_t})$ for a given pattern $\mathbf{e} = \mathbf{e}_{i_1 i_2 \ldots i_t}$, respectively. Our proof is based on the following observations:

(1) For a given pattern $\mathbf{e}$, $|\mathcal{F}_{\mathbf{e}}| + |\mathcal{N}_{\mathbf{e}}| = n + 1$ and HDT($\mathbf{e}, t$) = $\Pr(X_{\mathbf{e}} = 1) = \Pr(|\Delta_{\mathcal{F}_{\mathbf{e}}}| > |\Delta_{\mathcal{N}_{\mathbf{e}}}|)$.



(a) Two bit flips as a case of even t (b) Three bit flips as a case of odd t

Fig. 21. Position of flipping bits in challenge $\mathbf{c}$ and induced flipping ($\mathcal{F}$) and non-flipping zones ($\mathcal{N}$) in Φ: (a) flipping zones $\mathcal{F}$ and non-flipping zones $\mathcal{N}$ created by flipping bits $\mathbf{c}[i_1]$ and $\mathbf{c}[i_2]$, and (b) flipping zones $\mathcal{F}$ and non-flipping zones $\mathcal{N}$ created by flipping bits $\mathbf{c}[i_1]$, $\mathbf{c}[i_2]$ and $\mathbf{c}[i_3]$.

(2) If a pair of patterns $(\mathbf{e}, \mathbf{e}')$ with same t values satisfies $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$, then $\text{HDT}(\mathbf{e}', t) + \text{HDT}(\mathbf{e}, t) = 1$. The reason is that if $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$, then $\Pr(X_{\mathbf{e}} = 1) = \Pr(|\Delta_{\mathcal{F}_{\mathbf{e}}}| > |\Delta_{\mathcal{N}_{\mathbf{e}}}|) = \Pr(|\Delta_{\mathcal{N}_{\mathbf{e}'}}| > |\Delta_{\mathcal{F}_{\mathbf{e}'}}|) = 1 - \Pr(X_{\mathbf{e}'} = 1)$. Hence, $\text{HDT}(\mathbf{e}', t) = 1 - \text{HDT}(\mathbf{e}, t)$. Note that if $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$ holds, then $|\mathcal{N}_{\mathbf{e}'}| = |\mathcal{F}_{\mathbf{e}}|$.

(3) For a pair of patterns $(\mathbf{e}, \mathbf{e}'')$ with same t values, if $|\mathcal{F}_{\mathbf{e}''}| < |\mathcal{N}_{\mathbf{e}}|$ holds, then $\text{HDT}(\mathbf{e}'', t) + \text{HDT}(\mathbf{e}, t) < 1$. This fact can be proved as follows. Let us consider two pair of patterns $(\mathbf{e}, \mathbf{e}')$ and $(\mathbf{e}, \mathbf{e}'')$, and $|\mathcal{F}_{\mathbf{e}''}| < |\mathcal{F}_{\mathbf{e}'}|$ and $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$. Since $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$, $\text{HDT}(\mathbf{e}', t) = 1 - \text{HDT}(\mathbf{e}, t)$ holds as discussed above. If $|\mathcal{F}_{\mathbf{e}''}| < |\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$ holds, then we have:

$$\Pr(|\Delta_{\mathcal{F}_{\mathbf{e}''}}| > |\Delta_{\mathcal{N}_{\mathbf{e}''}}|) < \Pr(|\Delta_{\mathcal{F}_{\mathbf{e}'}}| > |\Delta_{\mathcal{N}_{\mathbf{e}'}}|) \tag{17}$$
$$\Rightarrow \text{HDT}(\mathbf{e}'', t) < \text{HDT}(\mathbf{e}', t) = 1 - \text{HDT}(\mathbf{e}, t) \Rightarrow \text{HDT}(\mathbf{e}'', t) + \text{HDT}(\mathbf{e}, t) < 1$$

We first consider the case of $\text{HDT}(t)$.

CASE-I: Odd t. Let us recall the definition of $\text{HDT}(t)$ from Eq. (5), i.e., $\text{HDT}(t) = \frac{1}{|\mathcal{E}_t|} \sum_{\mathbf{e} \in \mathcal{E}_t} \text{HDT}(\mathbf{e}, t)$, where $\mathcal{E}_t = \{\mathbf{e} : \text{HW}(\mathbf{e}) = t\}$. Now, we prove that $\text{HDT}(t) = 0.5$ for odd t. We start with developing a mechanism A_{odd} that finds a pattern $\mathbf{e}' \in \mathcal{E}_t$ for each $\mathbf{e} \in \mathcal{E}_t$ such that $\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t) = 1$. More formally, A_{odd} results following set $\mathcal{P}_t$ of pair of patterns: $\mathcal{P}_t = \bigcup\{(\mathbf{e}, \mathbf{e}') : |\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|\}$. We can rewrite the expression $\text{HDT}(t)$ based on the set $\mathcal{P}_t$ as:

$$\begin{aligned} \text{HDT}(t) &= \frac{1}{|\mathcal{E}_t|} \sum_{\mathbf{e} \in \mathcal{E}_t} \text{HDT}(\mathbf{e}, t) = \frac{1}{|\mathcal{E}_t|} \sum_{(\mathbf{e}, \mathbf{e}') \in \mathcal{P}_t} (\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t)) \\ &= \frac{1}{|\mathcal{E}_t|} \sum_{(\mathbf{e}, \mathbf{e}') \in \mathcal{P}_t} 1 = \frac{1}{|\mathcal{E}_t|} \times \frac{|\mathcal{E}_t|}{2} = \frac{1}{2} \end{aligned} \tag{18}$$

In case of $t = 1$, for the pattern $\mathbf{e} = \mathbf{e}_i$, there are only two intervals $\mathcal{I}_1 = \{0, \ldots, i\}$ and $\mathcal{I}_2 = \{i+1, \ldots, n\}$ as in Fig. 2a, and $\mathcal{F}_{\mathbf{e}} = \mathcal{I}_1$ and $\mathcal{N}_{\mathbf{e}} = \mathcal{I}_2$. For a given pattern $\mathbf{e}_i$, the mechanism A_{odd} defines a new pattern $\mathbf{e}_{i'}$, where $i' = n - i - 1$, and results in $\mathcal{F}_{\mathbf{e}'} = \mathcal{I}_1' = \{0, \ldots, n-i-1\}$, $\mathcal{N}_{\mathbf{e}'} = \mathcal{I}_2' = \{n-i, \ldots, n\}$, $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e}}|$ and $|\mathcal{N}_{\mathbf{e}'}| = |\mathcal{F}_{\mathbf{e}}|$. Hence, we have $\text{HDT}(1) = 0.5$ according to Eq. (18).

Now we consider the general case. When t is an odd number, for a pattern $\mathbf{e} = \mathbf{e}_{i_1, \ldots, i_t}$, there are $t+1$ intervals $\mathcal{I}_1, \ldots, \mathcal{I}_{t+1}$, and $\mathcal{F}_{\mathbf{e}} = \mathcal{A}_{i_1 \ldots i_t} = \mathcal{I}_1 \cup \mathcal{I}_3 \cup \ldots \cup \mathcal{I}_t$ and $\mathcal{N}_{\mathbf{e}} = \mathcal{B}_{i_1 \ldots i_t} = \mathcal{I}_2 \cup \mathcal{I}_4 \cup \ldots \cup \mathcal{I}_{t+1}$. Indeed, from these intervals, we can form pairs $(\mathcal{I}_1, \mathcal{I}_2), \ldots, (\mathcal{I}_{k-1}, \mathcal{I}_k), \ldots, (\mathcal{I}_t, \mathcal{I}_{t+1})$, $k = 2, \ldots, t+1$ and k is an even number. It is observed that the intervals $\mathcal{I}_{k-1}$ and $\mathcal{I}_k$ are separated by the index i_{k-1}, i.e., $\mathcal{I}_{k-1} = \{i_{k-2} + 1, \ldots, i_{k-1}\}$ and $\mathcal{I}_k = \{i_{k-1} + 1, \ldots, i_k\}$, and each pair $(\mathcal{I}_{k-1}, \mathcal{I}_k)$ forms a pair of flipping and non-flipping zones $(\mathcal{F}, \mathcal{N})$.

To build the desired mechanism A_{odd}, we exploit the idea as described for the case $t = 1$, i.e., for a given set of indices $\{i_1, \ldots, i_t\}$, the mechanism A_{odd} constructs the pattern $\mathbf{e}' = \mathbf{e}_{i_1', \ldots, i_t'}$ as follows:

(1) For all even k, we define $i_k' = i_k$, i.e., $i_2' = i_2, i_4' = i_4, \ldots, i_{t-1}' = i_{t-1}$.

(2) For each pair $(\mathcal{I}'_{k-1}, \mathcal{I}'_k)$, the index i'_{k-1} is chosen such that $|\mathcal{I}'_{k-1}| = |\mathcal{I}_k|$ and $|\mathcal{I}'_k| = |\mathcal{I}_{k-1}|$.

It is evident that for a given pattern $\mathbf{e} = \mathbf{e}_{i_1,\ldots,i_t}$ with $|\mathcal{F}_\mathbf{e}| = |\mathcal{A}_{i_1\ldots i_t}| = h$ and $|\mathcal{N}_\mathbf{e}| = |\mathcal{B}_{i_1\ldots i_t}| = n+1-h$, we can find a pattern $\mathbf{e}' = \mathbf{e}_{i'_1,\ldots,i'_t}$ with $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{A}_{i'_1\ldots i'_t}| = n+1-h = |\mathcal{N}_\mathbf{e}|$ and $|\mathcal{N}_{\mathbf{e}'}| = |\mathcal{B}_{i'_1\ldots i'_t}| = h = |\mathcal{F}_\mathbf{e}|$ by using the proposed mechanism A_{odd}. Hence, we proved that for each $\mathbf{e}_{i'_1,\ldots,i'_t}$ there exist a $\mathbf{e}'_{i'_1,\ldots,i'_t}$ such that $\text{HDT}(\mathbf{e}_{i'_1,\ldots,i'_t}, t) + \text{HDT}(\mathbf{e}_{i_1,\ldots,i_t}, t) = 1$, and $\text{HDT}(t) = 0.5$ for all t odd.

CASE-II: Even t. Similar to the case for odd value of t, we now construct an algorithm A_{even} which partitions the set $\mathcal{E}_t$ as: $\mathcal{P}_t = \bigcup\{(\mathbf{e}, \mathbf{e}') : |\mathcal{F}_{\mathbf{e}'}| < |\mathcal{N}_\mathbf{e}|\}$. Based on the set $\mathcal{P}_t$, we rewrite the expression $\text{HDT}(t)$ as:

$$\begin{aligned}\text{HDT}(t) &= \frac{1}{|\mathcal{E}_t|}\sum_{\mathbf{e}\in\mathcal{E}_t}\text{HDT}(\mathbf{e},t) = \frac{1}{|\mathcal{E}_t|}\sum_{(\mathbf{e},\mathbf{e}')\in\mathcal{P}_t}(\text{HDT}(\mathbf{e},t)+\text{HDT}(\mathbf{e}',t)) \qquad (19)\\ &< \frac{1}{|\mathcal{E}_t|}\sum_{(\mathbf{e},\mathbf{e}')\in\mathcal{P}_t}(\text{HDT}(\mathbf{e},t)+(1-\text{HDT}(\mathbf{e},t))) < \frac{1}{|\mathcal{E}_t|}\sum_{(\mathbf{e},\mathbf{e}')\in\mathcal{P}_t}1 = \frac{1}{|\mathcal{E}_t|}\times\frac{|\mathcal{E}_t|}{2} < \frac{1}{2}.\end{aligned}$$

We exploit the mechanism A_{odd} (developed for odd t), to develop an mechanism A_{even} for even values of t, such that for each pattern $\mathbf{e} \in \mathcal{E}_t$, we can find $\mathbf{e}' \in \mathcal{E}_t$ with $|\mathcal{F}_{\mathbf{e}'}| < |\mathcal{N}_\mathbf{e}|$, and $|\mathcal{N}_{\mathbf{e}'}| > |\mathcal{F}_\mathbf{e}|$. In this case, we have odd number of intervals, i.e., $\mathcal{I}_1, \ldots, \mathcal{I}_{t-2}, \mathcal{I}_{t-1}, \mathcal{I}_t, \mathcal{I}_{t+1}$ for a given pattern $\mathbf{e} = \mathbf{e}_{i_1,\ldots,i_t}$. From these intervals, we form pairs $(\mathcal{I}_1, \mathcal{I}_2), \ldots, (\mathcal{I}_{t-1}, \mathcal{I}_t)$ and the single unpaired interval $\mathcal{I}_{t+1}$. In this case with t flipping bits, $\mathcal{F}_\mathbf{e} = \mathcal{I}_2 \cup \mathcal{I}_4 \cup \cdots \cup \mathcal{I}_{t-2} \cup \mathcal{I}_t$, $\mathcal{N}_\mathbf{e} = \mathcal{N}_{\mathbf{e},1} \cup \mathcal{N}_{\mathbf{e},2}$, where $\mathcal{N}_{\mathbf{e},1} = \mathcal{I}_1 \cup \mathcal{I}_3 \cup \cdots \cup \mathcal{I}_{t-1}$ and $\mathcal{N}_{\mathbf{e},2} = \mathcal{I}_{t+1}$. Here, our objective is to find a pattern $\mathbf{e}' = \mathbf{e}_{i'_1,\ldots,i'_t}$ for a given pattern $\mathbf{e} = \mathbf{e}_{i_1,\ldots,i_t}$ which has $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e},1}|$, $|\mathcal{N}_{\mathbf{e}',1}| = |\mathcal{F}_\mathbf{e}|$. The indices $i'_1, \ldots, i'_t$ will be defined based on the $i_1, \ldots, i_t$, and pairs of intervals $(\mathcal{I}_1, \mathcal{I}_2), \ldots, (\mathcal{I}_{t-1}, \mathcal{I}_t)$ by employing the mechanism A_{odd}. Specifically, we set $i'_2 = i_2, i'_4 = i_4, \ldots, i'_t = i_t$, and $i'_1, i'_3, \ldots, i_{t-1}$ will be adjusted such that $(|\mathcal{I}'_1| = |\mathcal{I}_2|, |\mathcal{I}'_2| = |\mathcal{I}_1|), \ldots, (|\mathcal{I}'_{t-1}| = |\mathcal{I}_t|, |\mathcal{I}'_t| = |\mathcal{I}_{t-1}|)$.

By following this approach, we have $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e},1}|$ and $|\mathcal{N}_{\mathbf{e}',1}| = |\mathcal{F}_\mathbf{e}|$. Since $\mathcal{N}_\mathbf{e} = \mathcal{N}_{\mathbf{e},1} \cup \mathcal{N}_{\mathbf{e},2}$ and $|\mathcal{F}_{\mathbf{e}'}| = |\mathcal{N}_{\mathbf{e},1}|$, we have $|\mathcal{F}_{\mathbf{e}'}| < |\mathcal{N}_\mathbf{e}|$. It implies that $\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t) < 1$ (cf. Eq. (17)) and then $\text{HDT}(t) < 0.5$ (cf. Eq. (19)). We also have the following important observation: the gap between the $|\mathcal{N}_{\mathbf{e},1}|$ and $|\mathcal{N}_\mathbf{e}|$ reduces with decreasing value of $|\mathcal{N}_{\mathbf{e},2}|$. This implies that the gap between $|\mathcal{F}_{\mathbf{e}'}|(= |\mathcal{N}_{\mathbf{e},1}|)$ and $|\mathcal{N}_\mathbf{e}|$ reduces with decreasing value of $|\mathcal{N}_{\mathbf{e},2}| = |\mathcal{I}_{t+1}|$. To sum up, $\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t)$ approaches 1 with the decreasing value of $|\mathcal{N}_{\mathbf{e},2}| = |\mathcal{I}_{t+1}|$.

Now, we prove another fact: $\text{HDT}(t)$ approaches 0.5 with increasing t when t is even. Let us consider two even numbers t and t' where $t < t' \leq n$, and two patterns $\mathbf{a} = \mathbf{e}_{i_1,\ldots,i_t}$ and $\mathbf{b} = \mathbf{e}_{j_1,\cdots,j_{t'}}$, where the pattern $\mathbf{b}$ is defined as follows: the first t indices of pattern $\mathbf{b}$ are similar to that of pattern $\mathbf{a}$, i.e., $j_1 = i_1, \ldots, j_t = i_t$, and the remaining indices are $j_t < j_{t+1} < \cdots < j_{t'}$. By using the mechanism A_{even}, we can find patterns $\mathbf{a}'$ and $\mathbf{b}'$ corresponding to $\mathbf{a}$ and $\mathbf{b}$, respectively, where $A = \text{HDT}(\mathbf{a}, t) + \text{HDT}(\mathbf{a}', t) < 1$ and $B = \text{HDT}(\mathbf{b}, t') + \text{HDT}(\mathbf{b}', t') < 1$, respectively. Now, we show that $A < B < 1$. Since the pattern $\mathbf{a}$ has less number of 1's (representing the indices of flipping bits in challenge $\mathbf{c}$) than that of $\mathbf{b}$, it results $|\mathcal{N}_{\mathbf{a},2}| > |\mathcal{N}_{\mathbf{b},2}|$, i.e., value of $|\mathcal{I}_{t+1}|$ in case of $\mathbf{b}$ is smaller compared to pattern $\mathbf{a}$. In other words, $(|\mathcal{N}_\mathbf{a}| - |\mathcal{F}_{\mathbf{a}'}|) > (|\mathcal{N}_\mathbf{b}| - |\mathcal{F}_{\mathbf{b}'}|)$ holds. Since we have seen earlier that $A = \text{HDT}(\mathbf{a}, t) + \text{HDT}(\mathbf{a}', t) < 1$ when $|\mathcal{N}_\mathbf{a}| - |\mathcal{F}_{\mathbf{a}'}| \neq 0$ and $(\text{HDT}(\mathbf{a}, t) + \text{HDT}(\mathbf{a}', t)) \to 1$ when $(|\mathcal{N}_\mathbf{a}| - |\mathcal{F}_{\mathbf{a}'}|) \to 0$, we have following relation: $A = \text{HDT}(\mathbf{a}, t) + \text{HDT}(\mathbf{a}', t) < B = \text{HDT}(\mathbf{b}, t') + \text{HDT}(\mathbf{b}', t') < 1$. This addresses the fact that $\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t)$ approaches 1 with increasing value of t. Therefore, $\text{HDT}(t) = \frac{1}{|\mathcal{E}_t|}\sum_{(\mathbf{e},\mathbf{e}')\in\mathcal{P}_t}(\text{HDT}(\mathbf{e}, t) + \text{HDT}(\mathbf{e}', t))$ approaches 0.5 with increasing value of t.

Finally, we argue that $\mathrm{PC}(t)$ approaches 0.5 with increasing t. This follows from the fact that $\mathrm{PC}(t) = \frac{1}{t} \times \sum_{i=1}^{t} \mathrm{HDT}(i)$, $\mathrm{HDT}(i) \rightarrow 0.5$ for even $i \geq 4$ (cf. [11, Fig. 11]), and $\mathrm{HDT}(t) = 0.5$ for odd t. In general, $\mathrm{PC}(t)$ approaches 0.5 with increasing value of t. □

REFERENCES

[1] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. 2009. Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions. In *Proc. of ASIACRYPT*. 685–702.

[2] Georg T. Becker. 2015. On the Pitfalls of Using Arbiter-PUFs as Building Blocks. *IEEE TCAD* 34, 8 (2015), 1295–1307.

[3] Georg T. Becker. 2015. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In *Proc. of CHES*. 535–555.

[4] Christina Brzuska, Marc Fischlin, Heike SchrÃűder, and Stefan Katzenbeisser. 2011. Physically Uncloneable Functions in the Universal Composition Framework. In *Proc. of CRYPTO*. Vol. 6841. 51–70.

[5] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. 2014. Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?. In *Proc. of CHES*. 451–475.

[6] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC.

[7] Daihyun Lim. 2004. *Extracting Secret Keys from Integrated Circuits*. Master's thesis. MIT, USA.

[8] Roel Maes and Ingrid Verbauwhede. 2010. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*, Ahmad-Reza Sadeghi and David Naccache (Eds.). Springer, Berlin Heidelberg, 3–37.

[9] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. 2011. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. *IACR Cryptology ePrint Archive* 2011 (2011), 657.

[10] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. 2008. Lightweight secure PUFs. In *Proc. of IEEE/ACM ICCAD*. IEEE Press, Piscataway, NJ, USA, 670–673.

[11] M. Majzoobi, F. Koushanfar, and M. Potkonjak. 2008. Testing Techniques for Hardware Security. In *Proc. of IEEE International Test Conference(ITC)*. 1–10.

[12] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. 2009. Techniques for Design and Implementation of Secure Reconfigurable PUFs. *ACM Trans. Reconfigurable Technol. Syst.* 2, 1 (2009), 1–33.

[13] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. 2012. Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. In *Proc. of IEEE Symposium on Security and Privacy Workshops*. 33–44.

[14] Phuong Ha Nguyen and Durga Prasad Sahoo. 2016. An Efficient and Scalable Modeling Attack on Lightweight Secure Physically Unclonable Function. *IACR Cryptology ePrint Archive* 2016 (2016), 428.

[15] Phuong Ha Nguyen, Durga Prasad Sahoo, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. 2015. Efficient Attacks on Robust Ring Oscillator PUF with Enhanced Challenge-Response Set. In *Proc. of DATE*. 641–646.

[16] Phuong Ha Nguyen, Durga Prasad Sahoo, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. 2016. Security Analysis of Arbiter PUF and Its Lightweight Compositions Under Predictability Test. *ACM Trans. Des. Autom. Electron. Syst.* 22, 2, Article 20 (Dec. 2016), 28 pages. DOI: https://doi.org/10.1145/2940326

[17] Ravikanth S. Pappu. 2001. *Physical one-way functions*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[18] Bart Preneel, Werner Van Leekwijck, Luc Van Linden, René Govaerts, and Joos Vandewalle. 1990. Propagation Characteristics of Boolean Functions. In *Proc. of EUROCRYPT* . 161–173.

[19] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. 2013. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE TIFS* 8, 11 (2013), 1876–1891.

[20] Durga Prasad Sahoo, Phuong Ha Nguyen, Debdeep Mukhopadhyay, and Rajat Subhra Chakraborty. 2015. A Case of Lightweight PUF Constructions: Cryptanalysis and Machine Learning Attacks. *IEEE TCAD* 34, 8 (2015), 1334–1343.

[21] Durga Prasad Sahoo, Sayandeep Saha, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Hitesh Kapoor. 2014. Composite PUF: A New Design Paradigm for Physically Unclonable Functions on FPGA. In *IEEE HOST*. Arlington, VA, USA.

[22] G. Edward Suh and Srinivas Devadas. 2007. Physical unclonable functions for device authentication and secret key generation. In *Design Automation Conference*. ACM Press, New York, NY, USA, 9–14.

[23] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. 2014. Physical Characterization of Arbiter PUFs. In *Proc. of CHES*. 493–509.

[24] Arunkumar Vijayakumar and Sandip Kundu. 2015. A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics. In *Proc. of DATE*. 653–658.

[25] Meng-Day (Mandel) Yu, David M'Raïhi, Richard Sowell, and Srinivas Devadas. 2011. Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In *Proc. of CHES* , Vol. 6917. Springer Berlin / Heidelberg, 358–373.