Analysis of the Strict Avalanche Criterion in variants of Arbiter-based Physically Unclonable Functions

Akhilesh Anilkumar Siddhanti[1], Srinivasu Bodapati[2], Anupam Chattopadhyay[3], Subhamoy Maitra[4], Dibyendu Roy[5], and Pantelimon Stănică[6]

[1] Georgia Institute of Technology, Atlanta, USA
[2] Indian Institute of Technology Mandi, India
[3] Nanyang Technological University, Singapore
[4] Indian Statistical Institute, Kolkata, India
[5] ERTL(E), STQC, Kolkata, India
[6] Naval Postgraduate School, Monterey, USA
akhilesh@gatech.edu, srinivasu@iitmandi.ac.in, anupam@ntu.edu.sg, subho@isical.ac.in, roydibyendu.rd@gmail.com, pstanica@nps.edu

Abstract. Arbiter-based Physically Unclonable Functions (Arbiter PUF) were introduced to generate cryptographically secure secret keys during runtime, rather than storing it in Non-Volatile Memory (NVM) which are vulnerable to physical attacks. However, its construction was a target to several statistical and modeling attacks. One such statistical weakness of the Arbiter PUF is that it leaks information to the adversary, if some challenge-response pairs are known. The response is heavily biased towards the effect of flipping certain bits of the input, a widely studied property, known as the Strict Avalanche Criterion (SAC). Several variants of Arbiter PUFs have been proposed since then, with varying degrees of success against SAC. In this paper, we provide a generalized framework to analyze any Arbiter PUF variant against SAC. Building on this analysis, we propose a new Arbiter PUF variant which is not only highly resistant to SAC but also has very good reliability.

Keywords: Arbiter PUF, Boolean function, Bias, SAC.

1 Introduction

It is a well-known fact that storing cryptographic keys in Non-Volatile Memory (NVM) is insecure and is subject to physical attacks [7]. As an alternative, Arbiter-based Physically Unclonable Functions (PUFs) were introduced in [4], generating a huge interest in the cryptology community. An Arbiter-based PUF is a physical one-way function from an n-dimension space, referred to as *challenge* to a one bit output, called *response*. This function, when instantiated on a silicon chip, uses the inherent manufacturing process variations of the chip and transforms into an unknown Boolean function. This means that an instantiated

PUF cannot be cloned – even by the manufacturer. This has led to many interesting applications of Arbiter PUFs like identification and authentication [8,6,2], key generation and storage, and have even found their ways to commercially available products like RFID tags and smart cards [1]. But, there are some performance metrics an ideal Arbiter PUF should exhibit:
Uniqueness. When same challenges are given to different PUFs, the responses should be different as well. However, this does not happen in practice. There have been several works which show that when an Arbiter PUF is implemented on Xilinx Virtex-5/Kintex-7 FPGAs, the responses exhibit a very low value of uniqueness [12,10].
Uniformity. An Arbiter PUF should be uniform (or balanced), i.e., 50% of challenges should produce 0 as response, while the other 50% produce 1.
Reliability. An Arbiter PUF should produce the same response for a given input challenge. In practice, this fails due to several environmental variations, instabilities and aging in the circuit.

However, Arbiter PUFs have long been known to be susceptible to model building attacks using linear programming and ML techniques like Logistic Regression and SVM [15]. In a model building attack, the adversary develops a software model of the Physically Unclonable Function which gives the same response as output as the real PUF would generate (for the same challenge input). The model is built using the knowledge of several thousands of Challenge-Response pairs (CRPs) generated from the real PUF. Consequently, XOR [3] and Feedforward PUFs [8] were introduced. Several attacks have been reported on XOR PUFs since then, with [15] reporting attacks on up to 6-XOR PUFs, and several works have shown that an exponential increase in the number of challenge-response pairs (CRPs) is required for every additional XOR. Note that it has been reported in [18] that only a maximum of 12 XORs can be combined to provide a reliable response. However, it was recently shown in [1] that using a combination of reliability and CMA-ES, only a linear increase in CRPs is required to successfully clone XOR PUFs.

A statistical weakness prevalent in most of the Arbiter PUF variants has been reported in [14]. For an Arbiter PUF, there is a clear bias between the responses of two challenges which differ at 1 or more positions. This shows that the Boolean function, irrespective of the PUF construction, is *weak* in nature, since an Arbiter PUF can be distinguished from a random source by collection of very few samples. This gives a significant advantage to the adversary. This bias affects the XOR PUF and Feedforward PUF as well. In fact, we go on to show that the bias affects any general variant of Arbiter PUF. Based on this, we then develop a new variant of Arbiter PUF exhibiting very good properties of Strict Avalanche Criterion, while still maintaining reliability better than a XOR PUF with same number of Arbiter PUFs.

One of the first works to highlight a bias in the output for Arbiter-based PUFs was [14]. New Arbiter-based constructions were developed to mitigate bias and improve reliability [16], albeit under two different constructions, cMPUF and rMPUF. The authors in [16] also claim that the constructions withstand the

reliability-based attack presented by Becker in [1] and ML attacks like [15].
Our Contribution. The primary goal of this paper is to analyze several existing PUFs and propose an improved construction of PUF. An overview of our contribution is described below.

- We first discuss a theoretical derivation for a bias in an n-bit Arbiter PUF and n-bit k-XOR PUF (in Section 3).
- Next we discuss how the bias prevails in a Feedforward PUF construction (in Section 4).
- In Section 5, we provide a general framework to theoretically analyze bias and reliability of any Arbiter-based PUF construction.
- In Section 6 we introduce a new construction, S-PUF, and theoretically prove that the bias reduces significantly. We then combine several S-PUFs to build S_n-PUF with negligible bias and very high reliability and discuss its security against conventional machine learning attacks and Becker's reliability based attack [1] in Section 7, showing that it is the best PUF construction as far as we know.

2 Background

2.1 Description of Arbiter-based Physically Unclonable Functions and their variants

Using variations in a manufacturing process, Arbiter-based Physically Unclonable Functions generate a pseudorandom output based on an n-bit input. The input is commonly referred to as *challenge*, and 1-bit output as *response*. The function is said to be *reliable*, if it can produce the same response for every n-bit input. Several studies have discussed the reliability of Arbiter-based PUFs [18,2,16,1]. We are going to treat Arbiter-based PUFs from a Boolean function perspective, but before that we will explore the construction of Arbiter-based PUFs as known by the research community [3,8,15].
Arbiter PUF. A general Arbiter PUF consists of two symmetrically placed electrical paths consisting of n switches. A common pulse is transmitted at the same time on both the paths and received at the end by an arbiter. Every switch has a 1-bit input. If the input supplied to a switch is 0, the paths are not affected, whereas an input of 1 will swap the two paths (see Figure 1). Due to process variations, the pulse will traverse one path faster than the other. The arbiter simply generates the output as 0/1 depending on which pulse arrives first - the top or the bottom. Since the inception of Arbiter PUFs, they have been shown to be vulnerable against modeling attacks. One can model an arbiter PUF using machine learning techniques like Logistic Regression and Neural Networks [15], given enough challenge-response pairs. This prompted the introduction of XOR-based Arbiter PUFs and Feedforward PUFs.
k-XOR PUF. An k-XOR PUF combines the output of k Arbiter PUFs, each fed with the same challenge vector C, by a simple XOR (see Figure 2). The Arbiter PUF can be seen as a XOR PUF with $k = 1$.

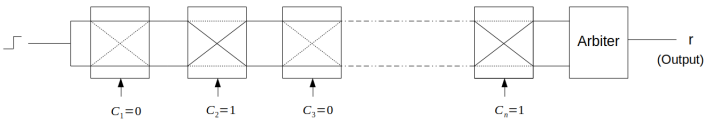


Fig. 1: An Arbiter PUF with n inputs.

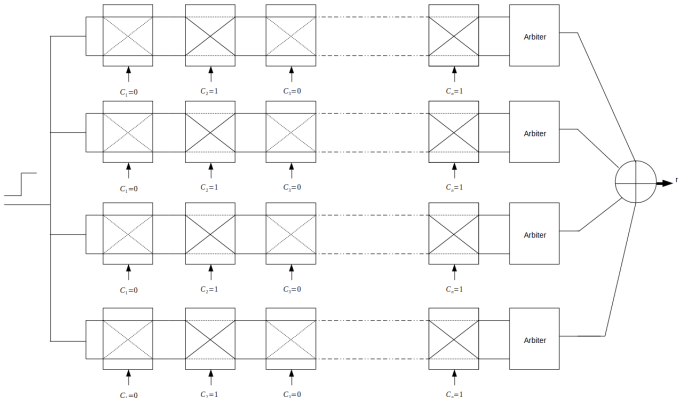


Fig. 2: k-XOR n-bit Arbiter PUF

Feedforward PUF. Another approach to construct Arbiter PUFs is to use additional arbiters as feedforward links [8]. The delay differences between the two paths after stage t_1 are checked using an arbiter, the output of which is fed as an input to an additional switch t_2. The input to the switch t_2 is denoted as C_{t_2}. The construction has been described in Figure 3.

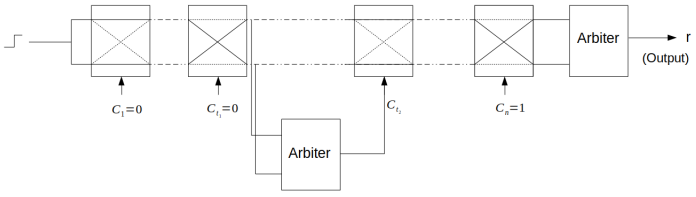


Fig. 3: Design of a Feedforward PUF

2.2 Modeling Arbiter-based PUFs as Boolean functions

Arbiter PUF. An Arbiter PUF can be represented as a function of the challenge and path delays due to the n switches. Here we follow the similar modeling as described in [9]. Let $\Delta(n)$ be the difference between the top and bottom paths of an n-length Arbiter PUF. It is customary to map the challenge and response bits using $f : \{0,1\} \rightarrow \{-1,1\}$ using the signature $f(x) = (1 - 2 \cdot x)$ for all $x \in \{C \cup r\}$. From here, we will assume $C_i \in \{-1,1\}$ and $r \in \{-1,1\}$ using the same mapping. The time taken by the pulse to traverse the two paths, *top* and

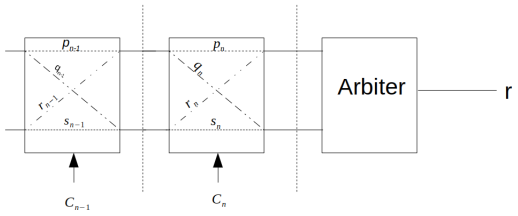


Fig. 4: The four possible delays a switch can introduce.

bottom, will be denoted as $\delta_{top}(n)$ and $\delta_{bottom}(n)$ respectively. At each switch i, four fixed, different values are introduced into the circuit, as shown in Figure 4. We denote these four delay values as p_i, q_i, r_i and s_i respectively. These p_i, q_i, r_i and s_i are randomly selected from a same normal distribution. So if the delay values $\delta_{top}(i)$ and $\delta_{bottom}(i)$ are known for some intermediate switch i, we can write $\delta_{top}(i+1)$ and $\delta_{bottom}(i+1)$ as:

$$\delta_{top}(i+1) = \frac{1+C_{i+1}}{2}(p_{i+1} + \delta_{top}(i)) + \frac{1-C_{i+1}}{2}(s_{i+1} + \delta_{bottom}(i)) \quad (1)$$

$$\delta_{bottom}(i+1) = \frac{1+C_{i+1}}{2}(q_{i+1} + \delta_{bottom}(i)) + \frac{1-C_{i+1}}{2}(r_{i+1} + \delta_{top}(i)). \quad (2)$$

The difference between the top and bottom paths is $\Delta(i)$. Subtracting Equations (1) and (2) will give us the following relation:

$$\Delta(i+1) = C_{i+1} \cdot \Delta(i) + \alpha_{i+1}C_{i+1} + \beta_{i+1}, \quad (3)$$

where $\alpha_i = \frac{(p_i - q_i)}{2} + \frac{(r_i - s_i)}{2}, \beta_i = \frac{(p_i - q_i)}{2} - \frac{(r_i - s_i)}{2}$. To simplify our expression, we denote the parity of challenge bits, P_k as: $P_k = \prod_{i=k+1}^{n} C_i$. By iteratively using integers $i \geq 0$ in Equation (3) and observing the pattern, the n^{th} term can be written as a function of parity bits and the delay values p_i, q_i, r_i and s_i:

$$\Delta(n) = \alpha_1 P_0 + (\alpha_2 + \beta_1)P_1 + \ldots + (\alpha_n + \beta_{n-1})P_{n-1} + \beta_n P_n, \quad (4)$$

where $P_k = \prod_{i=k+1}^{n} C_i$, for $k = 0, 1, \ldots, n-1$ and $P_n = 1$. Thus, an Arbiter PUF with n-bit input can be expressed as polynomial expression of the following form,

$\Delta(C) = \langle P, D\rangle$. Here P and D are the two vectors of $n+1$ length and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. The explicit form of P, D are $P = (P_0, P_1, \ldots, P_n)$, $D = (\alpha_1, \alpha_2 + \beta_1, \alpha_3 + \beta_2, \ldots, \alpha_n + \beta_{n-1}, \beta_n)$.

In the Equation (4) it can be observed that $P_i \in \{-1, 1\}$ for $i = 0, 1, \ldots, n-1$. Now if the sign of $\Delta(C)$ is negative the PUF outputs 0 and if sign of $\Delta(C)$ is positive then PUF outputs 1. We use the transformation $Q_i = \frac{1-P_i}{2}$ on Equation (4) for $i = 0, 1, \ldots, n-1$ and $Q_n = P_n$. After substituting $P_i = 1 - 2Q_i$ for $i = 0, 1, \ldots, n-1$ in Equation (4) we have the following.

$$\begin{aligned}\Delta(C) = \alpha_1(1-2Q_0) + (\alpha_2+\beta_1)(1-2Q_1) + (\alpha_3+\beta_2)(1-2Q_2) \\ + \cdots + (\alpha_n + \beta_{n-1})(1-2Q_{n-1}) + \beta_n Q_n.\end{aligned} \tag{5}$$

The response bit, r, is simply the sign value of $\Delta(n)$, that is, $r = sgn(\Delta(n))$. From Equation (5) it can be noticed that, after the above mentioned transformation, the PUF becomes a function which takes 0 or 1 as input and depending upon the sign of $\Delta(C)$ it outputs either 0 or 1. Complete PUF with n input can be seen as a Boolean function $f : \{0,1\}^n \to \{0,1\}$ involving n variables. The number of variables will be n as P_n is always a constant equal to 1, so the total number of variable Q_n is essentially n.

k-XOR PUF. The XOR PUF is equivalent to an arbiter PUF for $k = 1$. The response bit r can be represented as: $r = \bigoplus_{i=1}^{k} sgn(\Delta^i(n))$.

Feedforward PUF. A feedforward PUF uses an arbiter over the first t_1 stages, and feeds the input to the switch t_2 as C_{t_2}. We can represent this relation as: $C_{t_2} = sgn(\Delta(C_1, C_2, \ldots, C_{t_1}))$. We denote the response bit as: $z = sgn(\Delta(C_{\mathbf{x}}))$, where

$$\begin{aligned}\Delta(C_{\mathbf{x}}) = \alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n) + (\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n) \\ + \cdots + (\alpha_{t_1}+\beta_{t_1-1})(C_{t_1}\cdots C_{t_2}\cdots C_n) + \cdots + \\ + (\alpha_{t_2}+\beta_{t_2-1})(C_{t_2}\cdots C_n) + \cdots + (\alpha_n+\beta_{n-1})C_n + \beta_n\end{aligned} \tag{6}$$

3 Theoretical estimation of the experimental bias observed on r XOR PUF in [16]

In this section, we provide the exact theoretical estimation of the bias observed in [16]. We first recall the algebraic expression of the PUF which is described in Equation (4), namely $\Delta(C) = \langle P, D\rangle = \alpha_1 P_0 + (\alpha_2 + \beta_1)P_1 + \cdots + (\alpha_n + \beta_{n-1})P_{n-1} + \beta_n P_n$. Here, $P_k = \prod_{i=k+1}^{n} C_i \quad \text{for } k = 0, 1, \ldots, n-1$ and $P_n = 1$. In our model, we construct two inputs to the PUF by modifying the sign of C_1. We use $\Delta C_{\mathbf{x}}$, $\Delta C_{\widetilde{\mathbf{x}}}$ to denote ΔC corresponding to $\mathbf{x}$ and $\widetilde{\mathbf{x}}$ respectively. It can be noticed that all the monomials of $\Delta C_{\mathbf{x}}$ and $\Delta C_{\widetilde{\mathbf{x}}}$ have the same sign, except the first one $\alpha_1 P_0$, as C_1 is only involved in $\alpha_1 P_0$. It can be observed that $\Delta C_{\mathbf{x}}$

and $\Delta C_{\widetilde{\mathbf{x}}}$ will be of same sign if the following condition holds:

$$
\begin{aligned}
&|\alpha_1 P_0| < |(\alpha_2+\beta_1)P_1 + (\alpha_3+\beta_2)P_2 + \cdots + (\alpha_n+\beta_{n-1})P_{n-1} + \beta_n P_n| \\
&\Rightarrow |\alpha_1| < |(\alpha_2+\beta_1)P_1 + (\alpha_3+\beta_2)P_2 + \cdots + (\alpha_n+\beta_{n-1})P_{n-1} + \beta_n P_n| \quad (7) \\
&\Rightarrow |\alpha_1| < |X|, \quad \text{where } X = (\alpha_2+\beta_1)P_1 + (\alpha_3+\beta_2)P_2 \\
&\qquad\qquad + \cdots + (\alpha_n+\beta_{n-1})P_{n-1} + \beta_n P_n. \quad (8)
\end{aligned}
$$

Here, $\alpha_i = \frac{p_i - q_i}{2} + \frac{r_i - s_i}{2}$ and $\beta_i = \frac{p_i - q_i}{2} - \frac{r_i - s_i}{2}$ and p_i, q_i, r_i, s_i follows $\mathcal{N}(\mu, \sigma)$ (i.e., normal distribution with mean μ and standard deviation σ). As p_i, q_i, r_i and s_i follows the same normal distribution $\mathcal{N}(\mu, \sigma)$, so $p_i - q_i$, $r_i - s_i$ will follow $\mathcal{N}(0, \sqrt{2}\sigma)$. Hence α_i, β_i will follow $\mathcal{N}(0, \sqrt{2}\sigma)$. We use σ_1 to denote $\sqrt{2}\sigma$, so α_i and β_i will follow $\mathcal{N}(0, \sigma_1)$. From Equation (4) it can be seen that, $P_i \in \{-1, 1\}$, so $(\alpha_2+\beta_1)P_1 + (\alpha_3+\beta_2)P_2 + \ldots + (\alpha_n+\beta_{n-1})P_{n-1} + \beta_n P_n$ will follow $\mathcal{N}(0, \sqrt{2n-1}\sigma_1)$ i.e., $X \sim \mathcal{N}(0, \sqrt{2n-1}\sigma_1)$.

From Equation (8) we need to find the probability $Pr\Big[|\alpha_1| < |X|\Big]$. This is equivalent to find $Pr\Big[|\frac{\alpha_1}{X}| < 1\Big]$. In the following lemma we will find the required probability.

Lemma 1. *If $\alpha_1 \sim \mathcal{N}(0, \sigma_1)$ and $X \sim \mathcal{N}(0, \sqrt{2n-1}\sigma_1)$, then $Pr\Big[|\frac{\alpha_1}{X}| < 1\Big] = 1 - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)$.*

Proof. Here $\alpha_1 \sim \mathcal{N}(0, \sigma_1)$ and $X \sim \mathcal{N}(0, \sqrt{2n-1}\sigma_1)$. Firstly, we are interested to check the distribution of $\frac{\alpha_1}{X}$. For simplicity, we use σ_2 to denote $\sqrt{2n-1}\sigma_1$. Now, the probability density functions of the distribution of α_1 and X, denoted by $f_{\alpha_1}(\alpha_1)$ and $f_X(x)$, respectively, are given below $f_{\alpha_1}(\alpha_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{\alpha_1^2}{2\sigma_1^2}}$, $-\infty < \alpha_1 < \infty$; $f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{x^2}{2\sigma_2^2}}$, $-\infty < x < \infty$. We consider $\mathbf{y}_1 = \frac{\alpha_1}{X}$ and $\mathbf{y}_2 = X$, so $\alpha_1 = \mathbf{y}_1\mathbf{y}_2$ and $X = \mathbf{y}_2$. If $-\infty < \alpha_1, x < \infty$, then $-\infty < y_1, y_2 < \infty$. As the two distribution are mutually independent, the joint distribution of α_1, X will be $f_{\alpha_1,X}(\alpha_1, x) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left(\frac{\alpha_1^2}{2\sigma_1^2} + \frac{x^2}{2\sigma_2^2}\right)}$. The joint distribution of $\mathbf{y}_1, \mathbf{y}_2$ will be, $f_{\mathbf{y}_1,\mathbf{y}_2}(y_1, y_2) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left(\frac{y_1^2 y_2^2}{2\sigma_1^2} + \frac{y_2^2}{2\sigma_2^2}\right)} y_2$, where $-\infty < y_1, y_2 < \infty$. Now we are interested to find the distribution of $\mathbf{y}_1$ i.e., $f_{\mathbf{y}_1}(y_1)$, which is described below, $f_{\mathbf{y}_1}(y_1) = \int_{-\infty}^{\infty} f_{\mathbf{y}_1,\mathbf{y}_2}(y_1, y_2)dy_2 = \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left(\frac{y_1^2 y_2^2}{2\sigma_1^2} + \frac{y_2^2}{2\sigma_2^2}\right)} y_2 dy_2 = \frac{1}{\pi} \frac{\frac{\sigma_1}{\sigma_2}}{y_1^2 + (\frac{\sigma_1}{\sigma_2})^2}$. Our aim is to find the probability $Pr[|\frac{\alpha_1}{X}| < 1]$ i.e., for that we need to find $Pr[|\mathbf{y}_1| < 1]$. To find this, we do the following,

$$
\begin{aligned}
Pr[|\mathbf{y}_1| < 1] &= \Big| \int_{-1}^{1} \frac{1}{\pi} \frac{\frac{\sigma_1}{\sigma_2}}{y_1^2 + (\frac{\sigma_1}{\sigma_2})^2} dy_1 \Big| = \frac{1}{\pi} \Big| \{\tan^{-1}\left(\frac{1}{\frac{\sigma_1}{\sigma_2}}\right) - \tan^{-1}\left(\frac{-1}{\frac{\sigma_1}{\sigma_2}}\right)\} \Big| \\
&= \frac{2}{\pi}\tan^{-1}\left(\frac{\sigma_2}{\sigma_1}\right) = \frac{2}{\pi}\tan^{-1}(\sqrt{2n-1}) = 1 - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right),
\end{aligned}
$$

where $\sigma_2 = \sqrt{2n-1}\sigma_1$. Hence $Pr[|\frac{\alpha_1}{X}| < 1] = 1 - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)$. □

From the result of the Lemma 1 the following result follows.

Lemma 2. *Let* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *be two inputs to an* n *inputs PUF, where the first coordinates of* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *are of opposite in sign. Let* z *and* $\widetilde{z}$ *be the outputs corresponding to* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$*. Then* $Pr[z = \widetilde{z}] = 1 - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)$.

From the Lemma 2 we have $Pr[z = \widetilde{z}] = 1 - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right) = \frac{1}{2} + \left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)\right)$.

Now let us observe the bias for flipping t^{th} bit of the input. Revisiting Equation (4), we have (without any flips):

$$\begin{aligned}\Delta(C) &= \alpha_1(C_1 \cdots C_t \cdots C_n) + (\alpha_2 + \beta_1)(C_2 \cdots C_t \cdots C_n) + \cdots \\ &+ (\alpha_t + \beta_{t-1})(C_t \cdots C_n) + (\alpha_{t+1} + \beta_t)(C_{t+1} \cdots C_n) + \cdots \\ &+ (\alpha_n + \beta_{n-1})P_{n-1} + \beta_n P_n.\end{aligned}$$

Flipping the t^{th} bit results in the following expression:

$$\begin{aligned}\Delta(C) = -[&\alpha_1(C_1 \cdots C_t \cdots C_n) + (\alpha_2 + \beta_1)(C_2 \cdots C_t \cdots C_n) + \cdots \\ &+ (\alpha_t + \beta_{t-1})(C_t \cdots C_n)] + (\alpha_{t+1} + \beta_t)(C_{t+1} \cdots C_n) + \cdots \\ &+ (\alpha_n + \beta_{n-1})P_{n-1} + \beta_n P_n.\end{aligned}$$

We know that $\alpha \sim (0, \sigma_1\sqrt{2t-1})$ and $X \sim (0, \sigma_1\sqrt{2n-2t+1})$, where $\alpha = \alpha_1(C_1 \cdots C_t \cdots C_n) + (\alpha_2 + \beta_1)(C_2 \cdots C_t \cdots C_n) + (\alpha_t + \beta_{t-1})(C_t \cdots C_n)$ and X $= (\alpha_{t+1} + \beta_t)(C_{t+1} \cdots C_n) + \ldots + (\alpha_n + \beta_{n-1})P_{n-1} + \beta_n P_n$.

Thus, the bias can be written as:

$$Pr[z = \tilde{z}] = 1 - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}}\,. \tag{9}$$

Lemma 3. *For two inputs* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *differing in the* t^{th} *coordinate, we have* $Pr[z = \tilde{z}] = 1 - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}}$.

Further, we are interested in finding the bias of r-XOR PUFs. For that we use a Piling-up lemma, which is stated in Lemma 4 below.

Lemma 4. [13] *If* $Pr[X_i = 0] = p_i = \frac{1}{2} + \epsilon_i$*, for* $i = 1, \ldots, r$*, then* $Pr\Big[\bigoplus_{i=1}^{r} X_i = 0\Big] = \frac{1}{2} + 2^{r-1}\prod_{i=1}^{r}\epsilon_i$.

For a single PUF, we obtained that $Pr[z = \widetilde{z}] = \frac{1}{2} + \left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)\right)$ i.e., $Pr[z = \widetilde{z}] = \frac{1}{2} + \epsilon$, where $\epsilon = \left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)\right)$. Now if we consider r-XOR PUF with n inputs, then the bias of the same can be calculated from Lemma 4, which is stated in the following lemma.

Lemma 5. *Let* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *be two inputs to an* n *inputs* r*-XOR PUF, where the first coordinates of* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *are of opposite in sign. If* z *and* $\widetilde{z}$ *are the outputs corresponding to* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$*, then* $Pr[z = \widetilde{z}] = \frac{1}{2} + 2^{r-1}\epsilon^r$*, where* $\epsilon = \left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\left(\frac{1}{\sqrt{2n-1}}\right)\right)$.

This theoretical bias obtained in Lemma 5 is exactly the same as the experimental bias observed in [16] for several r XOR PUFs with n inputs.

4 Theoretical estimation of the bias in Feedforward PUF

Let us assume we have an n-bit feedforward PUF with a single feedforward link. The challenge vector is denoted by $C_{\mathbf{x}} = (C_1, C_2, \ldots, C_n)$. We will assume the input to the feedforward loop is taken after t_1 stages, and fed to switch t_2 so that $C_{t_2} = sgn(\Delta(C_1, C_2, \ldots, C_{t_1}))$. We denote the response bit as: $z = sgn(\Delta(C_{\mathbf{x}}))$

$$\begin{aligned}\Delta(C_{\mathbf{x}}) = {} & \alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n) + (\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)\\ & + \cdots + (\alpha_{t_1}+\beta_{t_1-1})(C_{t_1}\cdots C_{t_2}\cdots C_n) + \cdots +\\ & + (\alpha_{t_2}+\beta_{t_2-1})(C_{t_2}\cdots C_n) + \cdots + (\alpha_n+\beta_{n-1})C_n + \beta_n. \end{aligned} \tag{10}$$

We are interested in calculating the bias of $\Delta z = z \oplus \widetilde{z}$, where z corresponds to the input $\mathbf{x}$ and $\widetilde{z}$ corresponds to the input $\widetilde{\mathbf{x}}$. Here $\mathbf{x}$ and $\widetilde{\mathbf{x}}$ differ only at the first position. The following lemma proves the result.

Lemma 6. *Let* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *be two inputs to a feed forward PUF which takes feedback from the* t_1 *stage and feds to switch* t_2*. If the first coordinates of* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *are of opposite signs and the others are of the same sign, then*

$$\begin{aligned} Pr[z = \widetilde{z}] = {} & 1 - \frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}\\ & - \frac{4}{\pi^2}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}\tan^{-1}\frac{\sqrt{\frac{2t_2-2}{2n-2t_2+1}} - \frac{1}{\sqrt{2n-1}}}{1+\sqrt{\frac{2t_2-2}{(2n-2t_2+1)(2n-1)}}}, \end{aligned}$$

where z *and* $\widetilde{z}$ *correspond to the output bit corresponding to* $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$.

Proof. Flipping the first bit we have, $\widetilde{C}_1 = C_1 \times -1$ and $C_{t_2} = \Delta(C_1, C_2, \ldots, C_{t_1})$, where $\mathbf{x} = (C_1, C_2, \ldots, C_{t_2}, \ldots, C_n)$ and $\widetilde{\mathbf{x}} = (\widetilde{C}_1, C_2, \ldots, \widetilde{C}_{t_2}, \ldots, C_n)$. Hence, the sign of C_{t_2} will be different corresponding to $\mathbf{x}$ and $\widetilde{\mathbf{x}}$ as C_1 changes its sign. One may note that the sign of C_{t_2} will change with probability $\frac{2}{\pi}\tan^{-1}(\frac{1}{\sqrt{2t_1-1}})$ when the first bit C_1 changes its sign (see Lemma 2). We consider two cases.
Case 1 (The sign of C_{t_2} is altered). Here $\widetilde{C}_1 = -C_1$ and $\widetilde{C}_{t_2} = -C_{t_2}$. The Equation (10) becomes:

$$\begin{aligned}\Delta(C_{\mathbf{x}}) = {} & \alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n) - [(\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)\\ & + \cdots + (\alpha_{t_2}+\beta_{t_2-1})(C_{t_2}\cdots C_n)] + (\alpha_{t_2+1}+\beta_{t_2})(C_{t_2+1}\cdots C_n)\end{aligned}$$

$$+\cdots+(\alpha_n+\beta_{n-1})C_n+\beta_n.$$

It can be observed that the sign of $\Delta(C_{\mathbf{x}})$ and $\Delta(C_{\widetilde{\mathbf{x}}})$ will remain the same for $\mathbf{x}$ and $\widetilde{\mathbf{x}}$ iff $|\alpha|<|X|$, where $\alpha=(\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)+\cdots+(\alpha_{t_2}+\beta_{t_2-1})(C_{t_2}\cdots C_n)$ and $X=\alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)+(\alpha_{t_2+1}+\beta_{t_2})(C_{t_2+1}\cdots C_n)+\cdots+(\alpha_n+\beta_{n-1})\,C_n+\beta_n$. If $\alpha_i,\beta_i\sim\mathcal{N}(0,\sigma_1)$ then $\alpha\sim\mathcal{N}(0,\sigma_\alpha)$ and $X\sim\mathcal{N}(0,\sigma_X)$, where $\sigma_\alpha=\sigma_1\sqrt{2t_2-2}$, $\sigma_X=\sigma_1\sqrt{2n-2t_2+1}$. This leads to, $\frac{\sigma_\alpha}{\sigma_X}=\frac{\sqrt{2t_2-2}}{\sqrt{2n-2t_2+1}}$. Hence, under this scenario $Pr[z=\widetilde{z}|\text{Case 1 holds}]=1-\frac{2}{\pi}\tan^{-1}\frac{\sqrt{2t_2-2}}{\sqrt{2n-2t_2+1}}$ (see Lemma 2).

Case 2 (The sign of C_{t_2} is not altered). Here $\widetilde{C}_1=-C_1$ and $\widetilde{C}_{t_2}=C_{t_2}$. Now Equation (10) becomes:

$$\begin{aligned}\Delta(C_{\mathbf{x}})=&-[\alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)]\\&+(\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)+\cdots+(\alpha_{t_2}+\beta_{t_2-1})(C_{t_2}\cdots C_n)\\&+(\alpha_{t_2+1}+\beta_{t_2})(C_{t_2+1}\cdots C_n)+\cdots+(\alpha_n+\beta_{n-1})\,C_n+\beta_n.\end{aligned}$$

Here we have $\alpha=\alpha_1(C_1C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)$ and $X=(\alpha_2+\beta_1)(C_2\cdots C_{t_1}\cdots C_{t_2}\cdots C_n)+\cdots+\beta_n$. As we know $\alpha\sim\mathcal{N}(0,\sigma_1)$ and $X\sim\mathcal{N}(0,\sigma_1\sqrt{2n-1})$, $Pr[z=\widetilde{z}|\text{Case 2 holds}]=1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}$ (see Lemma 2).

It can be observed that $Pr[\text{Case 1 holds}]=\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}$ and $Pr[\text{Case 2 holds}]=1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}$. Combining Case 1 and Case 2 we have,

$$\begin{aligned}Pr[z=\widetilde{z}]&=Pr[z=\widetilde{z}|\text{Case 1 holds}]\times Pr[\text{Case 1 holds}]\\&\quad+Pr[z=\widetilde{z}|\text{Case 2 holds}]\times Pr[\text{Case 2 holds}]\\&=\left[\left(1-\frac{2}{\pi}\tan^{-1}\frac{\sqrt{2t_2-2}}{\sqrt{2n-2t_2+1}}\right)\left(\frac{2}{\pi}\tan^{-1}(\frac{1}{\sqrt{2t_1-1}}\right)\right]\\&\quad+\left[\left(1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}\right)\left(1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}\right)\right]\\&=1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}+\frac{4}{\pi^2}\tan^{-1}\frac{1}{\sqrt{2n-1}}\tan^{-1}\frac{1}{2t_1-1}\\&\quad-\frac{4}{\pi^2}\tan^{-1}\frac{\sqrt{2t_2-2}}{\sqrt{2n-2t_2+1}}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}\\&=1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}\\&\quad-\frac{4}{\pi^2}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}\left(\tan^{-1}\frac{\sqrt{2t_2-2}}{\sqrt{2n-2t_2+1}}-\tan^{-1}\frac{1}{\sqrt{2n-1}}\right)\\&=1-\frac{2}{\pi}\tan^{-1}\frac{1}{\sqrt{2n-1}}\\&\quad-\frac{4}{\pi^2}\tan^{-1}\frac{1}{\sqrt{2t_1-1}}\tan^{-1}\frac{\sqrt{\frac{2t_2-2}{2n-2t_2+1}}-\frac{1}{\sqrt{2n-1}}}{1+\sqrt{\frac{2t_2-2}{(2n-2t_2+1)(2n-1)}}}.\end{aligned}$$

The lemma is shown. □

5 Theoretical proof for a bias in any Arbiter PUF variant

In this section we will show several results on biases in any variant of Arbiter PUF. Here, we assume that the variant is constructed by placing k arbiters in parallel, each supplied by the same challenge input $C = \{C_1, \ldots, C_n\}$, whereas the outputs $r_1, r_2, \ldots, r_n$ are combined to give a singular output r by a combiner function f. First, we will develop a more general form of Piling-up Lemma that will give us a way to handle the weight of an Arbiter PUF.

Theorem 1. *Let $I = \{I_k\}_{k=1}^{s}$ be a partition of a set of n independent variables, say, $\{x_1, x_2, \ldots, x_n\} = I_1 \cup I_2 \cup I_3 \ldots \cup I_s$ such that $I_k \cap I_j = \phi$. Let $f : \{0,1\}^n \to \{0,1\}$, defined as $f = \sum_{k=1}^{s} \prod_{i \in I_k} x_i$ and $Pr[x_i = 0] = p_i = \frac{1}{2} + \epsilon_i$, where ϵ_i represents the bias in x_i. Then we have: $Pr(f(x_1, x_2, \ldots, x_n) = 0) = Pr\left[\sum_{k=1}^{s} \prod_{i \in I_k} x_i = 0\right] = \frac{1}{2} + 2^{s-1} \prod_{k=1}^{s} \epsilon_{I_k}$, where $\epsilon_{I_k} = P_{I_k} - \frac{1}{2}$, $P_{I_k} = 1 - \prod_{i \in I_k}(1 - p_i)$.*

Proof. We denote $Pr[x_i = 0] = p_i \quad \forall i \in \{1, \ldots, n\}$. Now since x_i and x_j are independent, $Pr[x_i x_j = 0] = 1 - Pr[x_i x_j = 1] = 1 - [Pr[x_i = 1] Pr[x_j = 1]] = 1 - [(1 - p_i)(1 - p_j)]$. Similarly,

$$\begin{aligned} Pr\left[\prod_{i \in I_k} x_i = 0\right] &= 1 - Pr\left[\prod_{i \in I_k} x_i = 1\right] = 1 - \prod_{i \in I_k} Pr[x_i = 1] \\ &= 1 - \prod_{i \in I_k} (1 - Pr[x_i = 0]) = 1 - \prod_{i \in I_k} (1 - p_i). \end{aligned}$$

In terms of bias, $Pr\left[\prod_{i \in I_k} x_i = 0\right] = 1 - \prod_{i \in I_k} (\frac{1}{2} - \epsilon_i)$.

We will now use mathematical induction on the sum of product terms to arrive at our result. Let us observe the behavior for the sum of two product terms:

$$\begin{aligned} &Pr\left[\prod_{i \in I} x_i + \prod_{j \in J} x_j = 0\right] \\ &= Pr\left[\prod_{i \in I} x_i = 0\right] Pr\left[\prod_{j \in J} x_j = 0\right] + Pr\left[\prod_{i \in I} x_i = 1\right] Pr\left[\prod_{j \in J} x_j = 1\right] \\ &= P_I P_J + (1 - P_I)(1 - P_J) = 1 - P_I - P_J + 2 P_I P_J = \frac{1}{2} + 2\epsilon_I \epsilon_J. \end{aligned}$$

Let us assume the following holds true for r product terms: $Pr\left[\sum_{j=1}^{r} \prod_{i \in I_j} x_i = 0\right] = \frac{1}{2} + 2^{r-1} \prod_{j=1}^{r} \epsilon_{I_j}$. For the $(r+1)^{th}$ term we have:

$$Pr\left[\sum_{j=1}^{r} \prod_{i \in I_j} x_i + \prod_{i \in I_r} x_i = 0\right]$$

$$
\begin{aligned}
&= Pr\left[\sum_{j=1}^{r}\prod_{i\in I_j} x_i = 0\right] Pr\left[\prod_{i\in I_r} x_i = 0\right] + Pr\left[\sum_{j=1}^{r}\prod_{i\in I_j} x_i = 1\right] Pr\left[\prod_{i\in I_r} x_i = 1\right] \\
&= \left(\frac{1}{2} + 2^{r-1}\prod_{j=1}^{r}\epsilon_{I_j}\right)\left(\frac{1}{2} + \epsilon_{I_{r+1}}\right) + \left(1 - (\frac{1}{2} + 2^{r-1}\prod_{j=1}^{r}\epsilon_{I_j})\right)\left(1 - (\frac{1}{2} + \epsilon_{I_{r+1}})\right) \\
&= \left(\frac{1}{2} + 2^{r-1}\prod_{j=1}^{r}\epsilon_{I_j}\right)\left(\frac{1}{2} + \epsilon_{I_{r+1}}\right) + \left(\frac{1}{2} - 2^{r-1}\prod_{j=1}^{r}\epsilon_{I_j})\right)\left(\frac{1}{2} - \epsilon_{I_{r+1}})\right) \\
&= \frac{1}{4} + 2^{r-2}\prod_{j=1}^{r}\epsilon_{I_j} + \frac{\epsilon_{I_{r+1}}}{2} + +2^{r-1}\prod_{j=1}^{r+1}\epsilon_{I_j} + \frac{1}{4} - 2^{r-2}\prod_{j=1}^{r}\epsilon_{I_j} - \frac{\epsilon_{I_{r+1}}}{2} + 2^{r-1}\prod_{j=1}^{r+1}\epsilon_{I_j} \\
&= \frac{1}{2} + 2^{r}\prod_{j=1}^{r+1}\epsilon_{I_j}.
\end{aligned}
$$

Hence, for s partitions on the set I, we have:
$Pr\left[\sum_{k=1}^{s}\prod_{i\in I_k} x_i\right] = \frac{1}{2} + 2^{s-1}\prod_{k=1}^{s}\epsilon_{I_k}$, and the theorem is shown. □

Corollary 1. *When $Pr[x_i = 0] = p$, for all x_i, $1 \le i \le n$, then $Pr[f(x_1, x_2, \ldots, x_n) = 0] = \frac{1}{2} + 2^{s-1}\prod_{k=1}^{s}\epsilon_{I_k}$, where $\epsilon_{I_k} = P_{I_k} - \frac{1}{2}, \quad P_{I_k} = 1 - (1-p)^{|I_k|}$.*

We next concentrate on finding the bias of the output of several PUFs combined via a Boolean function under the assumption that the probability of output of each PUF changes is constant p.

Theorem 2. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with the 0-support of f at $a \in \{0,1\}^n$ given by $\Omega_{f,a}^{(k)} = \{z \in \{0,1\}^n : \mathcal{C}_f(a) = -2^n + 2k\}$. Let $\Omega_{f,a}^{(k,w)} = \{z \in \Omega_{f,a}^{(k)} : wt(z) = k\}$. Further, we assume that the probability of any two input changes is $Pr[x_i + \tilde{x}_i = 0] = p$, for all i, where $\tilde{x}_i = x_i + a_i$, for $a_i \in \{0,1\}^n$. Then*
$Pr[f(x_1, \ldots, x_n) = f(\tilde{x}_1, \ldots, \tilde{x}_n)] = \frac{1}{2^n}\sum_{k=1}^{2^{n+1}}\sum_{w=0}^{n} kp^{n-w}(1-p)^w \left|\Omega_f^{(-2^n+2k,w)}\right|$.

Proof. We will start with an example to clear up the following argument. We take $f(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4 + x_1x_3x_4$, which has the autocorrelation spectrum $(16, 0, 0, 0, 8, 0, -8, 0, 8, 0, -8, 0, 8, 0, -8, 0)$ (the domain is ordered lexicographically). Therefore, taking into account the changes in the output for every possible value of a_i, where $\tilde{x}_i = x_i + a_i$, we see that for $a = (0,0,0,0)$, there are 16 values of x where $f(x) = f(\tilde{x})$, for $a = (0,0,0,1), (0,0,1,0), (0,0,1,1)$, $(0,1,0,1)$, $(0,1,1,1)$, $(1,0,0,1), (1,0,1,1), (1,1,0,1), (1,1,1,1)$ since the autocorrelation coefficient is 0, it means that there are 8 values of x (for each case), where $f(x) = f(\tilde{x})$, for $a = (0,1,0,0)$, $(1,0,0,0)$, $(1,1,0,0)$, since the autocorrelation coefficient is 8, there are 12 values of x, where $f(x) = f(\tilde{x})$ (and consequently, 4 values of x where $f(x) = f(\tilde{x}) + 1$), for $a = (0,1,1,0)$, $(1,0,1,0)$, $(1,1,1,0)$, since the autocorrelation coefficient is -8, there are 4 values of x,

where $f(x) = f(\tilde{x})$. Recall now that $Pr[x_i = \tilde{x}_i] = p$, and so partitioning the set of vectors a into vectors of the same weight whose autocorrelation coefficient is constant $2^4 - 2k \in \{-8, 0, 8\}$ (observe that for the autocorrelation coefficient -16 there are no values of x such that $f(x) = f(\tilde{x})$). Precisely,

$$\begin{aligned}\Omega_f =&\Omega_f^{(-8,2)} \cup \Omega_f^{(-8,3)} \cup \Omega_f^{(-8,2)} \cup \Omega_f^{(0,1)} \cup \Omega_f^{(0,2)} \cup \Omega_f^{(0,3)} \cup \Omega_f^{(0,4)} \cup \Omega_f^{(8,1)} \\ &\cup \Omega_f^{(8,2)} \cup \Omega_f^{(16,0)},\end{aligned}$$

where the first label represents the autocorrelation coefficient and the second is the weight of the vectors, for example $\Omega_f^{(-8,2)} = \{(0,1,1,0),(1,0,1,0)\}$, we get

$$\begin{aligned}&16 \cdot Pr[f(x_1,x_2,x_3,x_4) + f(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,\tilde{x}_4)] \\ &= 4 \cdot p^2(1-p)^2 \left|\Omega_f^{(-8,2)}\right| + 4 \cdot p(1-p)^3 \left|\Omega_f^{(-8,3)}\right| + 8 \cdot p^3(1-p) \left|\Omega_f^{(0,1)}\right| \\ &+ 8 \cdot p^2(1-p)^2 \left|\Omega_f^{(0,2)}\right| + 8 \cdot p(1-p)^3 \left|\Omega_f^{(0,3)}\right| + 8 \cdot (1-p)^4 \left|\Omega_f^{(0,4)}\right| \\ &+ 12 \cdot p^3(1-p) \left|\Omega_f^{(8,1)}\right| + 12 \cdot p^2(1-p)^2 \left|\Omega_f^{(8,1)}\right| + 16 \cdot p^4 \left|\Omega_f^{(16,0)}\right| \\ &= 40p^3(1-p) + 44p^2(1-p)^2 + 28p(1-p)^3 + 8(1-p)^4 + 16p^4.\end{aligned}$$

Now, for the general case, we let, for $k \geq 1$ (recall that the autocorrelation coefficients are all even) $\Omega_f^{(-2^n+2k,w)} = \{a : \mathcal{C}_f(a) = -2^n + 2k, wt(a) = w\}$, and reasoning exactly as in the example above, we get the probability

$$Pr[f(x_1,\ldots,x_n) = f(\tilde{x}_1,\ldots,\tilde{x}_n)] = \frac{1}{2^n} \sum_{k=1}^{2^n} \sum_{w=0}^{n} kp^{n-w}(1-p)^w \left|\Omega_f^{(-2^n+2k,w)}\right|.$$

We will now conduct experiments and verify Theorem 2 on various combiner functions.

5.1 Experimental verification with variants of Arbiter PUFs.

We have verified Theorem 2 for various combiner functions and few of them are mentioned in this section. The experiments have been conducted using C programming. One may note that theoretical parameters do not depend on the delay parameters (i.e., distribution parameters), so for our experiments we have considered standard pseudorandom generators like lrand(), srand().

PUF Type	Combiner Function and parameters	Experimental Bias	Theoretical Bias (Theorem 2)
k-XOR	$f = x_1 + x_2 + x_3 + x_4$, $n = 10$ and $k = 4$	0.630	0.629
(n,k)-MPUF [16]	$f = x_1x_5x_6 + x_2x_5x_6 + x_2x_5 + x_3x_5x_6$ $+x_3x_6 + x_4x_5x_6 + x_4x_5 + x_4x_6 + x_4$ $n = 10$ and $k = 2$	0.759	0.760

6 S-PUF construction: Improving the SAC property

In this section, we propose a new Arbiter-based PUF construction to improve its SAC property, using only two Arbiter PUFs. The construction not only shows a significant improvement in the SAC property over other PUF constructions but also has a reliability of an 2-XOR PUF. We denote this new construction as S-PUF, since it involves a circular 'shift' of the challenge vector.

The construction involves two n-bit Arbiter PUFs, where the input $C = \{C_1, C_2, \ldots, C_n\}$ of the first Arbiter PUF is shifted by $n/2$ positions and fed to the second Arbiter PUF as the input:

$$\tilde{C} = \{C_{n/2}, C_{n/2+1}, \ldots, C_n, C_1, \ldots, C_{n/2-1}\}, \tag{11}$$

where $\tilde{C}$ is the input to the second Arbiter. The output of two PUFs is simply XORed to produce the output.

6.1 Design rationale

The motivation arises from Lemma 3, where we try to find the value of t with minimum bias: $Pr[z = \tilde{z}] = 1 - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}} = \frac{1}{2} + \frac{1}{2} - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}}$ $= \frac{1}{2} + \epsilon'$, where $\epsilon' = \frac{1}{2} - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}}$. Putting $\epsilon' = 0$ we have: $\epsilon' = \frac{1}{2} - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}} = 0 \implies \tan^{-1}\sqrt{\frac{2t-1}{2n-2t+1}} = \frac{\pi}{4} \implies t = \lceil\frac{n+1}{2}\rceil$ or $\lfloor\frac{n+1}{2}\rfloor$. For even n we have experimentally observed that $t = \lfloor\frac{n+1}{2}\rfloor$ i.e., $t = \frac{n}{2}$ provides the lowest bias. For odd n the bias will be lowest for $t = \frac{n+1}{2}$. As in general n is even, we assume $t = \frac{n}{2}$.

For a k-XOR PUF, when a challenge bit is flipped, the same bias is observed in every participant Arbiter PUF contributing to the XOR. However, in the S-PUF construction we move the flip position to a different location for the second Arbiter PUF to reduce the bias as much as possible. Note that since it is not possible to move every bit flip to $(\frac{n}{2})$ in the second Arbiter PUF, we choose the next best available position for the same. This construction comes with no additional requirement of hardware or memory, as shown in Figure 5.

6.2 Bias in the S-PUF construction

Now let us observe the bias in the S-PUF, as proposed. Using the Piling-up Lemma and Equation (3), we have:

$$\begin{aligned} Pr[z = \tilde{z}] &= \frac{1}{2} + \epsilon_1\epsilon_2 \\ &= \frac{1}{2} + \left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t_1-1}{2n-2t_1+1}}\right)\left(\frac{1}{2} - \frac{2}{\pi}\tan^{-1}\sqrt{\frac{2t_2-1}{2n-2t_2+1}}\right) \\ &= \frac{1}{2} + \epsilon'. \end{aligned} \tag{12}$$

We take $t_2 = ((t_1 + (\frac{n}{2}) - 2) \bmod n) + 1$.

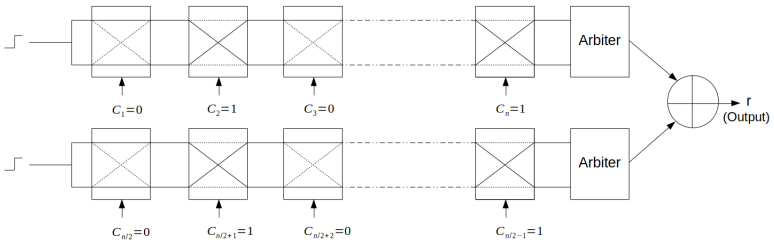
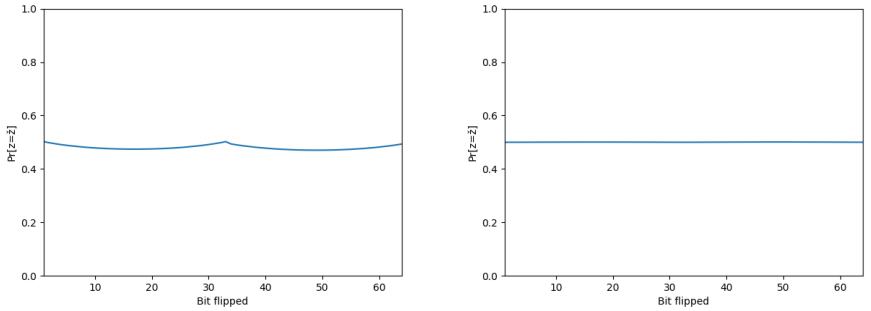


Fig. 5: S-PUF construction with only two Arbiter PUFs



(a) Bias for S-PUF construction involving only two Arbiter PUFs

(b) Bias for S-PUF construction involving 4 Arbiter PUFs

7 Constructions of S_n-PUFs

An S-PUF construction consisting of 2 Arbiter PUF has very good SAC property, but is very weak against model-building attacks. An S-PUF will have a similar level of security as that of a 2-XOR PUF against linear programming and ML attacks. Hence, we need to use multiple Arbiter PUFs to fend off model building attacks. We will describe a special combiner function which will take inputs from several S-PUFs and produce a 1-bit response. Combined together, we call this new PUF construction, an S_n-PUF. Therefore, an S_n-PUF is composed of n S-PUFs (where each S-PUF is as mentioned in Section 6). For the following combiner functions: $f_1(x) = x_1 + x_2x_3 + x_4x_5x_6 + x_7x_8x_9x_{10} + x_{11}x_{12}x_{13}x_{14}x_{15}$, $f_2(x) = x_1x_2x_3x_4x_5 + x_6x_7x_8x_9x_{10} + x_{10}x_{11}x_{12}x_{13}x_{14}x_{15}$, $f_3(x) = x_{14} + x_{15} + x_1x_2x_3x_4x_5 + x_6x_7x_8x_9x_{10} + x_{10}x_{11}x_{12}x_{13}$, the reliabilities have been described in Table 1. We notice that the functions mentioned in Table 1 provide good reliabilities. We also observe that other cryptographic properties of the functions f_1, f_2 and f_3 are not very good. We mention some cryptographic properties of these function in Table 2. Note that the functions f_1, f_2 and f_3 are either highly biased towards 0 or do not have good nonlinearity even if they produce good reliability. We need a function which has good nonlinearity and produces

Arbiters used	Combiner Function	Reliability of single PUF	Overall Reliability
30	f_1	0.96	0.87
30	f_2	0.96	0.97
30	f_3	0.96	0.89

Table 1: Reliability for S-PUF with combiner

Function	Variables	Nonlinearity	Bias
f_1	15	11344	0.5
f_2	15	2404	0.073
f_3	15	3664	0.5

Table 2: Cryptographic properties of combiner functions

reasonable good reliability. We observed that the M-M (Maiorana-McFarland) bent functions [11] which have the highest nonlinearity produce reasonable good reliability. The general algebraic normal form of an M-M bent function involving $2n$ number of variables is $f(x, y) = \phi(x) \cdot y + g(x)$, where $x, y \in \{0, 1\}^n$, $\phi(y)$ is a permutation and g is a Boolean function on n variables. For our combiner function we assume $g(y) = 0$ (as it provides good reliability), i.e., the combiner function is $f(x, y) = \phi(x) \cdot y$. In Table 3 we provide reliabilities of different S-PUFs, when we consider $f(x, y) = \phi(x) \cdot y$ as the combiner function. Here, our prime assumption is that the input to the combiner function must be even.

Arbiters used	Input to the combiner	Reliability	Nonlinearity	Bias
24	12	0.81	2016	0.4922
28	14	0.78	8128	0.4960
32	16	0.76	32640	0.4980

Table 3: S-PUF with $f(x, y) = \phi(x) \cdot y$ as combiner function

7.1 Security discussions on S_n-PUFs

All Arbiter-based PUFs are susceptible to model-building attacks through Machine Learning techniques like LR and SVM [15]. In [1], it was shown that only a linear increase in the number of CRPs is required to build a robust model to cryptanalyze XOR PUFs. Keeping these results in mind, we stand in the favor of S_n-PUFs using a two-fold argument:

Security against Conventional Machine Learning Attacks. Note that current ML techniques (apart from Becker's reliability attack reported in [1]) have only been able to attack up to 6-XOR PUFs (as reported in [15]). Hence, we choose S_{12} PUF (involving 24 Arbiter XOR PUFs) to be sufficiently safe against any modern machine learning attack known to us. A 24-XOR PUF will have very low reliability, while S_{12}-PUF performs very well in terms of reliability (see Table 3) and the SAC property (see Section 6).

Security against Becker's Attack in [1]. Becker's attack uses the randomness of CMA-ES algorithm to converge at a solution. The attacks proceeds with

measuring reliability values of each PUF in the n-XOR PUF as $h_0, h_1, \ldots, h_{n-1}$ and generating a random n-XOR PUF model and measuring the reliability of each of its constituent PUFs as $\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_{n-1}$. Note that we cannot obtain reliability values of constituent PUFs in a n-XOR PUF directly. For this, the author explains in [1, Section 5.2]: *"Let us assume for a PUF model $\overrightarrow{\omega}$ and for a challenge $\overrightarrow{\Phi_i}$ of one of the n Arbiter PUFs, the expected reliability is low, i.e., $\tilde{h} = 0$. Then the measured reliability h_i should also be low, since a bit flip of one of the response bits that are XORed directly results in a bit flip of the output of an XOR PUF. Hence, in this case the measured reliability h_i matches with the computed reliability $\tilde{h_i}$. If the computed reliability for the challenge Φ_i is high, i.e., $\tilde{h_i} = 1$, and the observed reliability is also high, the computed and observed reliability vectors match each other."* Note that a "bit flip" in any constituent Arbiter PUFs for an S_n-PUF does not mean a necessary effect in the output, due to the nature of the M-M type construction. Thus the reliability parameters cannot be obtained to mount the attack. Further, the reliability of each PUF is not equally represented in the output reliability of S_n-PUF due to high nonlinearity of the combiner function, making it even more difficult to model. Combining this with several factors such as the ability to use as large number of Arbiter PUFs (increasing n) is possible while maintaining very good reliability (PUFs exhibiting higher reliability make Becker's attack even more difficult), S_n-PUF can defend reliability based attacks quite well if not completely circumvent it.

8 Experimental results

The proposed S-PUF is implemented in Verilog with placement constraints. The designed PUF is tested on Nexys-4 DDR Artix-7 FPGA board. The responses for a given challenge to the PUF are recorded using a serial communication UART protocol. Next, we will discuss about the performance metrics of the PUF and the implications of those in deriving a security key.

The main functionality of a PUF is it should be able to generate all possible combinations of the output in the response. In an ideal case, for a PUF with 16-bit challenge should be able to produce different 2^{16} combinations of outputs. That means each challenge should have a unique response. To validate this behavior, the *Hamming distance* of the response is plotted to check the distribution of the response. A test case of challenges with a one-bit flip in consecutive challenges are given to the PUF and the responses are recorded. This Hamming distance is also known as the *Intra Hamming distance* and can be expressed as $\textit{Intra Hamming distance} = \sum_{i=1}^{k} \frac{HD(R_i, R_{i+1})}{n} \times 100$; here, k is the total number of challenges given to the PUF, R_i and R_{i+1} are the responses to the challenges C_i and C_{i+1}, respectively. Figure 6 is the plot displaying the Hamming distance of the responses with the iterations. The plot is like a Gaussian plot with a maximum at the half of its response length. Moreover, the plot shows that the proposed S-PUF is able to generate different combinations of the responses over its length.

The response from the PUF is mostly used as a security key or as an encryption key. The key must have few properties such as the key must be uniformly distributed in nature and it should be noise free. Based on these requirements, the few parameters for the PUF are defined as *uniqueness, uniformity, bit-aliasing and reliability.*

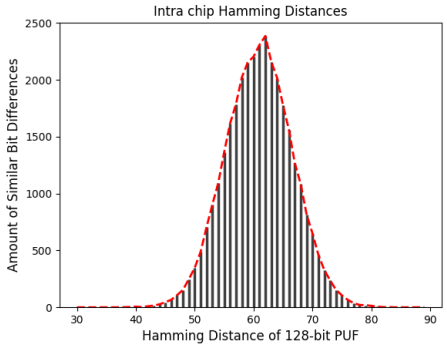


Fig. 6: Hamming distance of a 128-bit S-PUF on a Nexys-4 DDR Artix-7 FPGA

Uniqueness. The physical significance of uniqueness is how uniquely the PUF can distinguish its output from one device to the other of same configuration. That is, the same PUF design on two different chips (or devices) of the same configuration should not give the same output. The uniqueness of the PUF design is evaluated using the inter chip Hamming distance and the same is expressed as $Uniqueness = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i,R_j)}{n} \times 100$. Here, k is the number of chips considered for experiment. The inter chip Hamming distance is calculated as the Hamming distance between R_i and R_j, where R_i and R_j are the responses of the PUF to the same challenge C from two different chips (FPGA) $FPGA_i$ and $FPGA_j$. The ideal value of uniqueness is 50%, that is, half of the bits should be different. From the plot it is clear that the proposed S-PUF has a maximum Inter Hamming distance at half of its response length, therefore, the uniqueness of the PUF is 50%. Uniqueness is calculated over two FPGAs of same family.

Uniformity. The response of the PUF should satisfy the condition of having an equal number of 0's and 1's in the response, to use it as a security key. The uniformity of a chip is calculated as $Uniformity = \frac{1}{n} \sum_{k=1}^{n} R[k] \times 100$. Here, k is the bit position and n is the length of the response. The ideal value of the uniformity is 50%. Since all possible responses are considered to calculate the average, the value might deviate slightly. The maximum, minimum and average values of the uniformity can be considered, while evaluating the design.

Bit-aliasing. The physical significance of the bit-aliasing parameter is whether any particular bit position in the response is permanently sourced to either 0 or 1. The response bit must be derived from the applied challenge and the physical properties of the circuit. When bit-aliasing happens, different chips produce the same response irrespective of the challenge given. The bit aliasing of a particular

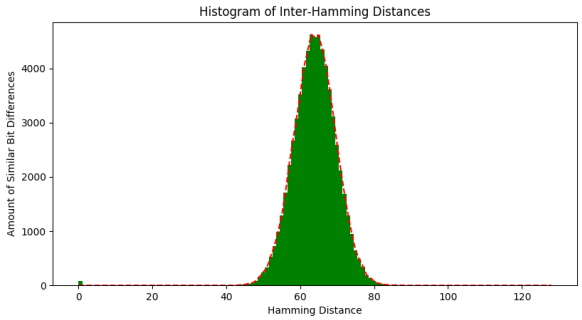


Fig. 7: Uniqueness of a 128-bit S-PUF on two Nexys-4 DDR Artix-7 FPGA's with same challenge set

bit position over different chips is calculated as $bit\text{-}alising_p = \frac{1}{n}\sum_{i=1}^{k} R_i[p]$. Here, k is the number of chips used for experiment, $R_i[p]$ is the p^{th}-bit of the response (R_i) of chip i. The ideal value of bit−aliasing is 50%, which indicates the equal distribution of 0 and 1.

Reliability. Reliability is one of the main concerns in today's electronic devices and systems. Similarly, the reliability of the PUF is an important metric to use PUF as a security primitive. The significance of the reliability of PUF translates to how efficiently a PUF can reproduce the same response bits under different situations. The situations can be the aging effect, the heating of the device or variations in the voltage levels of the output. Since, today's electronic circuits are more prone to the noise, any variation in the response bits will not serve the purpose of the PUF:

$$IHD = \frac{1}{m}\frac{\sum_{t=1}^{m} HD(R_i, R_i^t)}{n} \times 100 \tag{13}$$

$$reliability = (100 - IHD)\%. \tag{14}$$

The reliability of the PUF is calculated by using the *intra Hamming distance* of the response for same challenge given for several iterations. The reliability can be calculated from the Equations (13) and (14); here, n is the length of the response R_i and m is the total number of iterations. The ideal value of the reliability is 100%. Usually, for a good PUF design, the reliability is varied by a few error bits.

From the plot shown in Figure 8, it can be observed that the reliability of the PUF is very good. The maximum error in a response is about 20 bits, which can be recovered using some error correction mechanism. The error occurred can be easily recovered using an error correction code scheme such as BCH, or Reed-Solomon codes.

8.1 Comparison of the S-PUF results

The comparison of the PUF metrics studied above are summarized here with few existing designs. The designs presented in [5] have two variations one with

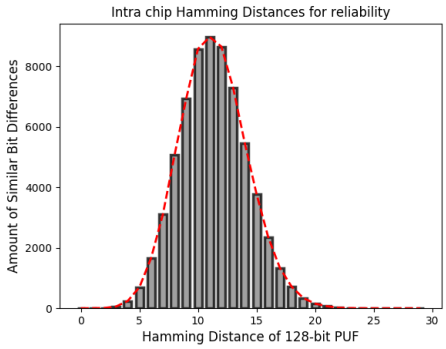


Fig. 8: Reliability of a 128-bit S-PUF on a Nexys-4 DDR Artix-7 FPGA

the post characterization(CHAR) and other one including the error correction capability (CHAR & MAJ). Table 4 summarizes the metrics and it is evident that the proposed S-PUF has a good performance.

Parameter	Proposed S-PUF	Ring oscillator PUF [17]	Original	CHAR	CHAR & MAJ
			Design [5]		
Uniqueness	50 %	49.22	48.52	45.60	45.60
Uniformity	49.9%	48.5	51.06	50.60	50.54
Reliability	92 %	99.99	92.00	98.87	99.58
Bit-aliasing	51.7 %	47.89	43.52	43.52	43.52

Table 4: Comparison of the S-PUF with few existing designs; Reliability of the design is without ECC while the other designs with ECC

9 Conclusion

In this paper, we have provided a theoretical estimation of the previously observed experimental bias. We noticed that these biases do not depend on the parameters of the distribution. These theoretical proofs of biases are also experimentally verified by doing several experiments in hardware, as well as in software. From our theoretical estimates, the biases of several PUFs with large inputs can be determined, which is not possible, computationally. Further, we introduce a new construction of a PUF, which overcomes these biases. We want to point out that our new construction is secure against existing attack techniques. The proposed S-PUF is implemented in an Artix-7 FPGA and we observe that the S-PUF has a good uniqueness, bit-aliasing and uniformity. The design also has a good reliability of 92%, where the error can be recovered using some error correction mechanism.

Acknowledgments. We would like to thank the anonymous reviewers of Indocrypt 2019 for their valuable suggestions and comments, which considerably improved the quality of our paper.

References

1. Georg T. Becker, The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 535–555. Springer, Berlin, Heidelberg, 2015.
2. Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. Secure lightweight entity authentication with strong PUFs: Mission impossible?. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 451–475. Springer, Berlin, Heidelberg, 2014.
3. Srini Devadas. Physical unclonable functions and secure processors. In: Cryptographic Hardware and Embedded Systems-CHES 2009, pp. 65-65. Springer, Berlin, Heidelberg, 2009. Available online at: `https://link.springer.com/chapter/10.1007/978-3-642-04138-9_5`
4. Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In: Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160. ACM, 2002. Available online at: `https://dl.acm.org/citation.cfm?id=586132`
5. Chongyan Gu, Neil Hanley and Máire O'neill. Improved Reliability of FPGA-Based PUF Identification Generator Design. ACM Trans. Reconfigurable Technol. Syst. Vol. 10, No. 3, 20:1–20:23, July 2017.
6. Ghaith Hammouri, and Berk Sunar. PUF-HB: A tamper-resilient HB based authentication protocol. In: International Conference on Applied Cryptography and Network Security, pp. 346-365. Springer, Berlin, Heidelberg, 2008.
7. Oliver Kömmerling, and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. Smartcard 99 (1999): 9–20. Available online at: `http://static.usenix.org/events/smartcard99/full_papers/kommerling/kommerling.pdf`
8. Jae W. Lee, Daihyun Lim, Blaise Gassend, G. Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In 2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525), pp. 176–179. IEEE, 2004. Available online at: `https://people.csail.mit.edu/devadas/pubs/vlsi-symp-puf.pdf`
9. Daihyun Lim, Daihyun Lim, Blaise Gassend, G. Edward Suh, Marten Van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 13 (10), pp. 1200–1205, 2005.
10. Takanori Machida, Dai Yamamoto, Mitsugu Iwamoto, and Kazuo Sakiyama. A new mode of operation for arbiter PUF to improve uniqueness on FPGA. In: 2014 Federated Conference on Computer Science and Information Systems, pp. 871–878. IEEE, 2014.
11. James A. Maiorana. A class of bent functions. R41 Technical Paper. August 1970.
12. Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In: Embedded systems design with FPGAs, pp. 245–267. Springer, New York, NY, 2013.
13. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, ISBN: 0-8493-8523-7, October 1996.
14. Phuong Ha Nguyen, Durga Prasad Sahoo, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. Security analysis of arbiter PUF and its lightweight com-

positions under predictability test. ACM Transactions on Design Automation of Electronic Systems (TODAES) 22, no. 2 (2017): 20.
15. Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In: Proceedings of the 17th ACM conference on Computer and communications security, pp. 237–249. ACM, 2010. Available online at:`https://eprint.iacr.org/2010/251.pdf`
16. Durga P. Sahoo, Debdeep Mukhopadhyay, Rajat S. Chakraborty and Phuong H. Nguyen. A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security. IEEE Transactions on Computers, Vol. 67, No. 3, 2017, 403–417.
17. B. Srinivasu, P. Vikramkumar, Anupam Chattopadhyay, Kwok-Yan Lam. CoLPUF: A Novel Configurable LFSR-based PUF. In: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2018, 358–361.
18. Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. IEEE Transactions on Multi-Scale Computing Systems 2, no. 3 (2016): 146–159.