Machine Learning Attacks on Low-Cost Reconfigurable XRRO and XRBR PUF Designs

Manthan Kojage, Neelofar Hassan, and Urbi Chatterjee(✉)

Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India
{manthank,neelofar,urbic}@cse.iitk.ac.in

Abstract. Physically unclonable functions (PUFs) can be seen as hardware circuits whose output does not only depend upon the inputs fed to it, but also on the random variation in the integrated circuits (ICs) during its manufacturing process. As a result of their unique hardware fingerprinting, these circuits can be used to authenticate devices among a population of identical silicon chips, much like a human being can be authenticated by their biometrics. In ACM TECS 2019, two low-cost reconfigurable Strong PUF designs namely XOR-based Reconfigurable Bistable Ring PUF (XRBR PUF) and XOR-based Reconfigurable Ring Oscillator PUF (XRRO PUF) have been proposed as a promising low-cost solution for IoT security. The two notable features of these architectures are: i) both of them exploit the logic reconfigurability which is efficient in terms of hardware cost, and ii) they exhibit good uniqueness and reliability properties. These make XRRO and XRBR PUFs good candidates for Strong PUF-based authentications and an interesting target for the machine learning (ML) adversaries as the machine learning resiliency was never discussed for both the cases in the proposal. In this paper, we develop a mathematical model for both of the designs by exploiting a common flaw of not having any non-linear component in the structure. Hence they are proven to be as vulnerable as their forerunner designs such as Configurable Ring Oscillator PUF and Bistable Ring PUFs. Finally, we show through experimental analysis that 128-bit XRBR PUFs can be broken with $10K$ CRPs with an accuracy of approximately 99%. On the other hand, for 127-stage XRRO PUFs having $8, 16, 32, 64$ layers of XRROs can be broken with $200K, 1M, 3M, 8M$ CRPs with an accuracy of approximately 97%–99%.

Keywords: Physically unclonable functions · Machine learning · XOR gate · Bistable ring · Configurable Ring Oscillator

1 Introduction

Over the past two decades, Physically Unclonable Functions (PUFs) have garnered significant attention from the research community worldwide [6,14]. They

L. Batina et al. (Eds.): SPACE 2022, LNCS 13783, pp. 204–224, 2022.
https://doi.org/10.1007/978-3-031-22829-2_12

have been employed in a number of authentication protocols as building blocks. As a component of hardware-based security primitive, PUFs create a distinct digital fingerprint for a circuit or device using manufacturer process variations. PUFs cannot be physically replicated because of its unpredictable, small-scale manufacturing variances. Even the manufacturer is unable to deliberately duplicate a PUF instance. As a result, producing physically identical specimens is infeasible. PUFs use a challenge-response mechanism to serve as hardware primitive. We provide the PUF with *"challenge"* (denoted as C), which act as a trigger or excitation signal, causing them to react by producing a *"response"* (denoted as R_C). The produced response depends on the physical characteristics of the PUF and the challenge fed to it. The challenge fed, and the response generated by the PUF is called the challenge-response pair (CRP) of the PUF. Based on the size of the challenge-response space of the PUF circuit, it can classified in two categories, namely Weak PUFs and Strong PUFs.

- **Weak PUFs:** Weak PUFs [7,8,22,23], are generally used for on-device secret key generation. It is generally assumed that the response generated by this kind of PUFs never leave the hardware platform.
- **Strong PUFs:** Strong PUFs [4–6,16,25] possess huge challenge-response pairs which are mainly used for device authentication. It is not feasible to construct all the 2^N challenge-response pairs (CRPs) for N-bit Strong PUF and to search the particular response for any arbitrary challenge in polynomial time. In the state-of-the-art literature, the Strong PUFs architectures such as Arbiter PUF (APUF), XOR-Arbiter PUF (XOR-APUF), Configurable Ring Oscillator PUF (CRO PUF) and Bistable Ring PUF (BR PUF) have proven to be versatile cryptographic primitives with a wide set of applications, such as key establishment and identification protocols [18]. However, the Strong PUFs suggested in the literature consume significant hardware resources, and thus, are not scalable for lightweight applications in IoT framework. To address this problem, recently two lightweight PUF architectures such as XOR-based Reconfigurable Ring Oscillator PUF (XRRO PUF) and XOR-based Reconfigurable Bistable Ring PUF (XRBR PUF) [13] have been proposed which incur lesser hardware overhead while retaining good uniqueness and reliability values.

But, due to the emergence of classical and reliability-based machine learning (ML) attacks, most of the PUF compositions are proven to be vulnerable in recent state-of-the-art literature [3,17,19–21,24]. Ruhrmair *et al.* proposed the first ML attacks against strong PUFs in 2010 [19,20], based on a mathematical delay model [12] of the APUF, RO PUF, XOR-APUF, Feed Forward APUF, and LSPUF. A similar kind of delay-based modelling is done for CRO PUF in which the delays of the oscillation of the CROs are exploited [17]. Memory-based Strong PUFs such as Bistable Ring PUF has also been modelled in which the pull strengths of the logic gates are exploited [24]. Several machine learning techniques, such as Support Vector Machine (SVM), Logistic Regression (LR), and Covariance Matrix Adaptation Evolution Strategies(CMA-ES), have been used to demonstrate these models [11,19]. Hence, while assessing the quality

of a Strong PUF candidate, it is of paramount importance that we not only estimate the uniqueness, uniformity and reliability properties of the design but also evaluate the robustness of the circuit against ML-based attacks.

Hence, in this paper, we try to answer the question that: *Though XRRO and XRBR PUFs are suitable for generating hardware fingerprints for lightweight applications, how vulnerable are they against ML-based modelling attacks?* The intuition for modelling these PUF architectures is that they inherit the similar flaws as CRO PUF and BR PUF that make them vulnerable against the ML attacks [17,24]. Though the vulnerabilities are similar, but due to the change in the construction of XRRO and XRBR PUF, the exact model for CRO PUF and BR PUF might not be directly applied to these two architectures. Hence the main crux of this work is to use the knowledge of the mathematical model formation of the former and build the same specifically for the XRRO and XRBR PUF. To the best of our knowledge, no prior works have been done to perform a security analysis on XRRO and XRBR constructions so far.

Overall, this paper's main contributions can be summed up as follows:

- We examine the security vulnerability and machine learning resiliency of XRRO and XRBR PUF architecture which was missing in the actual proposal [13]. We propose a mathematical model of the same to successfully launch ML-based attacks.
- We also implement the attack on simulated XRRO and XRBR PUF and evaluate the security architectures against SVM and LR algorithms. We further show the efficiency of the proposed attack by successfully modeling both the designs with 97%–99% accuracy.

The rest of the paper is organized in the following manner. In Sect. 2 we provide the background related to the proposed mathematical model and the machine learning attacks. The novel mathematical model proposed in this paper is illustrated in Sect. 3. The machine learning attacks and the experimental results are presented in Sect. 4. Finally, we conclude the paper with Sect. 5.

2 Background

In this section, we first discuss the basic working principles of Configurable Ring Oscillator PUF (CRO PUF) [16] and Bistable Ring PUF (BR PUF) [4]. Then we briefly discuss mathematical models that are used to perform machine learning attacks on these architectures.

2.1 CRO PUFs and BR PUFs

The basic building block of CRO PUF is a Ring oscillator PUF (RO PUF). The RO PUF is generally made of an identically laid-out loop of an odd number of logically inverting delay elements (as shown in Fig. 1). An RO generally continues to oscillate between 0 and 1 when triggered because the output of the last buffer is always the logical "NOT" of the input fed. An RO PUF [22] utilizes this

non-settling property of RO to introduce randomness in the circuit and it can be exploited for hardware fingerprinting. Figure 2 shows the circuit diagram of a traditional RO PUF. There are two multiplexers MUX1 and MUX2, each of which has N selection lines. There are 2^N ROs that are connected to both these multiplexers. The oscillating frequency of each oscillator in RO PUF is unique across devices due to the manufacturing process variations and can not be predicted apriori. The challenges are divided into two parts and provided to the select lines of multiplexers of RO PUF respectively in order to select a pair of ROs, and then, their frequencies are calculated using the counters. Finally, these two counter values are compared to generate an output. If RO selected by MUX1 has a higher oscillation frequency, the output is 1, otherwise, the output is 0.

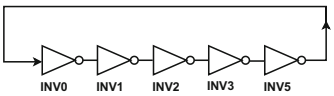


Fig. 1. Traditional 5-stage RO.

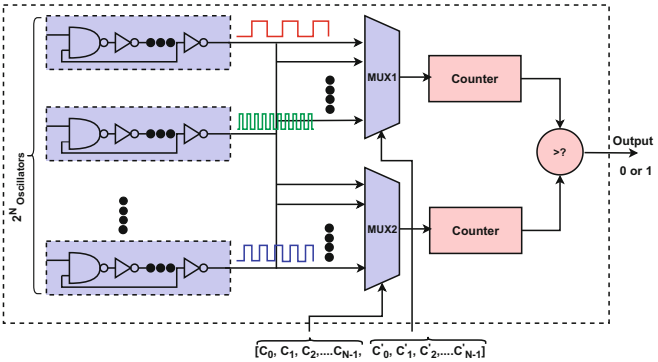


Fig. 2. Ring Oscillator PUF [22].

Now, from the implementation perspective, it has been found that if we increase the challenge space, the number of ROs need to be added to the circuit grows exponentially. Hence the hardware overhead will be very high if we try to use RO PUF as a *Strong PUF candidate*. On the other hand, the RO PUFs have higher reliability compared to other Strong PUF architectures [15]. To balance this trade-off, the Configurable Ring Oscillator PUF(CRO PUF) circuit [16] has been proposed as a variant of the RO PUF. We show its circuit design as well in Fig. 3. Here each RO can be reconfigured by using a multiplexer to select one of two inverters at each stage. The selection bit (S_i) for the multiplexer at the i^{th} stage acts as a configurable signal which determines whether the upper or lower

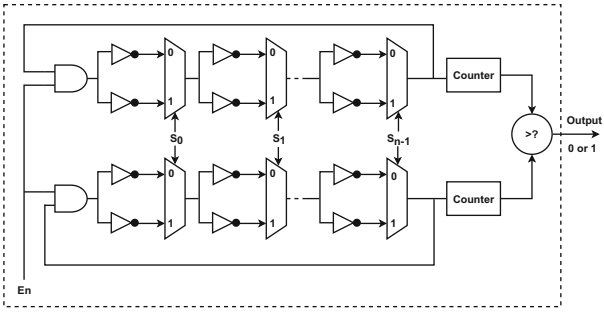


Fig. 3. The architecture of the CRO PUF [16].

delay element will be used at that stage. Finally, the output frequencies of the two CROs are compared to derive the PUF response. The main advantage of using CRO PUF is that we can do away with 2^N layers of ROs as in RO PUF (please ref to Fig. 2). Hence it is more efficient in terms of hardware overhead and can be considered as a Strong PUF as well.

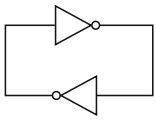


Fig. 4. Schematic diagram of a SRAM cell [8].

On the other hand, BR PUF is comprised of a cycle of an even number of logically inverted delay elements. Its main principle is the same as that of SRAM PUF [9] i.e. it exploits the unstable properties of SRAM cell transience. Whenever it is powered on, every SRAM cell (as shown in Fig. 4) results in invariably random deviations in the threshold voltages. This variation is extracted in the startup values of "uninitialized" SRAM memory. Consequently, an SRAM response produces a unique, random binary bit patterns. Now, A n-bit BR PUF (as shown in Fig. 5) is composed of n stages, where each stage has two inverting delay elements (e.g. NOR gates). The challenge bits $\{C_0, C_1, ..., C_{n-1}\}$ applied to the multiplexer and demultiplexer of every stage selects the NOR gates used in each bistable ring configuration. Once triggered, the BR PUF works like a memory cell and will enter either "101010..." or "010101..." as one of its two stable states. An n-bit BR PUF can have a total of 2^n different configurations as each NOR gate has a distinct process variation and each challenge vector generates a distinct bistable ring configuration.

Before allowing the ring to stabilise and produce a response, a synchronous RESET signal is applied to every stage in order to start with an all-0 state. When RESET is low and the ring starts to oscillate through the selected NOR gates,

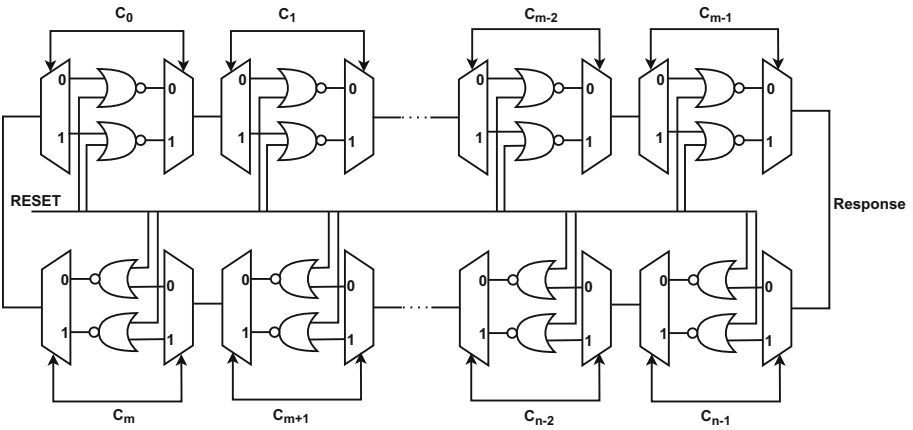


Fig. 5. Schematic diagram of a n-bit BR PUF [4,24]

the response is evaluated. Once the ring reaches a stable state, the outputs of any two neighboring stages will be opposite to each other. In the ring configuration, the noise and process variation of the NOR gates are utilised to determine which of the two probable stable states of the ring is selected. Moreover, an output port can be created at any node that connects two stages.

We next briefly discuss the mathematical models of CRO and BR PUFs.

2.2 ML Attacks on CRO PUFs and BR PUFs

ML-based modeling attacks on Strong PUF candidates are very prominent in the state-of-the-art literature. The adversarial model that is assumed over here is that a small subset of the challenge-response pairs (CRPs) are given to the adversary for a particular PUF design. The hurdle is whether (s)he can build a mathematical model of the same using the given CRPs to achieve high prediction accuracy while guessing responses for an unknown challenge.

First we briefly discuss the mathematical model of the CRO PUF. Each stage of a CRO PUF consists of four delay components: $(\delta_{i1}, \delta_{i2})$ for the upper and lower delay line of the top CRO (as shown in Fig. 6) and $(\delta'_{i1}, \delta'_{i2})$ for the upper and lower delay line of the bottom CRO. Now the oscillation frequency of the top and bottom CROs will be decided by the configuration of the paths through which the signal propagates and their delay contributes to the overall summation. And the choice of paths depends on the challenge bit selection. If the upper CRO is faster, i.e., the oscillation frequency is more than the lower CRO, we get a response of 1; otherwise, the response is 0. Now, the delay difference for every stage can be formulated as given below [17]:

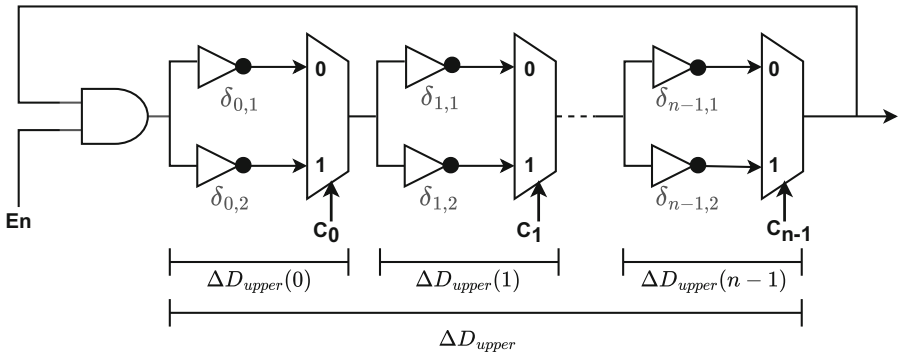


Fig. 6. CRO PUF diagram showing delay components of upper CRO [17].

$$\Delta D(i) = \Delta D_{upper}(i) - \Delta D_{lower}(i) \tag{1}$$

$$\begin{aligned} \Delta D_{upper}(i) &= \frac{1-C_i}{2}\delta_{i1} + \frac{1+C_i}{2}\delta_{i2} \\ \Delta D_{lower}(i) &= \frac{1-C_i}{2}\delta'_{i1} + \frac{1+C_i}{2}\delta'_{i2} \end{aligned} \tag{2}$$

$$\Delta D(i) = \frac{1-C_i}{2}(\delta_{i1} - \delta'_{i1}) + \frac{1+C_i}{2}(\delta_{i2} - \delta'_{i2}) \tag{3}$$

Please note that for i-th stage of the CRO, the challenge bits $C_i \in \{-1, 1\}$ is bipolar-encoded. Let,

$$\delta_i^{\alpha} = \delta_{i1} - \delta'_{i1}$$
$$\delta_i^{\beta} = \delta_{i2} - \delta'_{i2}$$

Now similar to the mathematical modelling of Arbiter PUF [21], the overall delay difference between the oscillation of upper and lower CRO for an n-stage CRO-PUF can be described as a linear sum of vector dot products.

$$\Delta D = \sum_{i=0}^{n-1} \Delta D(i) = \overrightarrow{P_{\alpha}}.\overrightarrow{W_{\alpha}} + \overrightarrow{P_{\beta}}.\overrightarrow{W_{\beta}} \tag{4}$$

where,

$$\overrightarrow{P_{\alpha}} = \left\{ \frac{1-C_0}{2}, \frac{1-C_1}{2} ..., \frac{1-C_{n-1}}{2} \right\}$$

$$\overrightarrow{P_{\beta}} = \left\{ \frac{1+C_0}{2}, \frac{1+C_1}{2} ..., \frac{1+C_{n-1}}{2} \right\}$$

$$\overrightarrow{W_{\alpha}} = \left\{ \delta_0^{\alpha}, \delta_1^{\alpha}, ..., \delta_{n-1}^{\alpha} \right\}, \overrightarrow{W_{\beta}} = \left\{ \delta_0^{\beta}, \delta_1^{\beta}, ..., \delta_{n-1}^{\beta} \right\}$$

This model can be used to launch CMA-ES and LR based ML attacks on the CRO PUF [17]. We have used the similar idea in Sect. 3 to model XRRO PUF as proposed in [13].

Similarly, an additive model has been put forth in the literature for predicting the resolution of metastability [10], with weights representing the strength with which different cells pull toward a particular outcome. Using this idea, a mathematical model of the BR PUF has been made [24]. Each gate in the BR PUF has weight associated with it representing difference between the pull-up strength and pull-down strength of the same. The overall response given a specific challenge(C) is determined by the summation of the weights across all the gates in accordance with the path that the applied challenge has selected. A positive sum indicates that the configured ring provides a "1", whereas a negative sum indicates that ring provides "–1" value. As the pull-up strength of even stages and the pull-down strength of odd stages favour a overall positive response, the summation of weights requires negative and positive polarities.

Let t_i and b_i represent the difference in the pull-up and pull-down strength of the top and bottom NOR gates in the i^{th} stage respectively (please ref to Fig. 5).

The total strength pulling toward the positive response for a given n-bit challenge is the summation of n number of t_i or b_i weights (depending on whether C_i is $+1$ or -1) and can be given as [24],

$$R_C = sgn(\sum_{i=0}^{n-1} \left(-1^i.\frac{t_i - b_i}{2}\right) + C_i \left(-1^i.\frac{t_i + b_i}{2}\right) \tag{5}$$

where C_i is the challenge bit of the i^{th} stage and is bipolar-encoded i.e. $C_i \in \{-1, 1\}$. The sign of R_C could be used to predict the response. For convenience, α_i and β_i can be defined as $\left(-1^i.\frac{t_i - b_i}{2}\right)$ and $\left(-1^i.\frac{t_i + b_i}{2}\right)$ respectively. Then,

$$R_C = sgn(\sum_{i=0}^{n-1} \alpha_i + C_i.\beta_i) \tag{6}$$

Given that the weights α_i and β_i are not known and since a BR PUF has only two possible responses, The response prediction of a BR PUF can be converted into a classification problem based on the given equation. We have exploited similar notion in Sect. 3 to model XRBR PUF.

3 Modelling XOR-Based Reconfigurable PUFs

In this section we first briefly describe the working principle of XRBR PUF and XRRO PUF followed by our proposed mathematical models for the same.

To start with, XOR gate is the basic building block for both the designs. It has a property that it can act as both buffer and inverter relying on the input values. The mechanism of XOR-based Reconfigurable PUFs [13] is based the above property of XOR gates. Let A and B be the inputs to an XOR gate. When B = "0", the XOR gate acts as a buffer relaying the input value A to the

output following a delay. When B = "1", the XOR gate's output value is the logic inverse of the input A, hence it acts as an inverter. Thus by controlling one of its bits, an XOR gate can be switched from a buffer to an inverter and vice-versa.

3.1 Mechanism of XRBR PUF

As discussed in Sect. 2.1, every stage of BR PUF consists of two inverting delay elements. The gate is chosen in each bistable ring configuration by providing the challenge bit at the select line of the MUX and DEMUX of that particular stage (please refer to Fig. 5). The XRBR PUF replaces each stage of BR PUF with an XOR gate (as shown in Fig. 7). The stages of XRBR PUF thus can be configured as buffers or inverters by controlling one of the inputs to the XOR gate of that stage. The selection bit is provided to each stage of a XRBR PUF to configure it into a unique circuit. As the number of stages in the BR PUF is even which makes it act like a memory circuit, The XOR gates configured as inverters in the XRBR PUF must be even in number. It implies that the configuration bits to the XRBR PUF should have an even number of 1's, which in turn configures an even number of XOR gates as inverters befitting it to behave like a memory circuit similar to BR PUF. Now there are variations due to the hardware intrinsic properties such as delay and driving capabilities of the XOR gate for every buffers and inverters of the XRBR PUF. This property is absolutely similar to SRAM PUF as the impurities are induced in the platform due to the random process variations in the manufacturing phase. Additionally, different configuration of the XRBR PUF adds to the variation in the output to the circuit and results in different responses of the PUF instance.

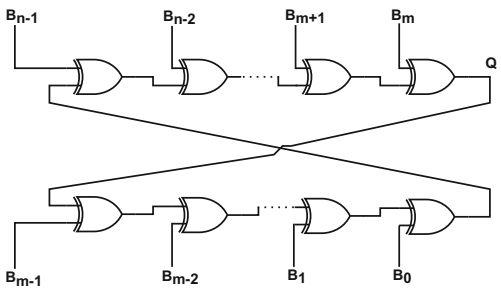


Fig. 7. The schematic diagram of an n-bit XRBR PUF [13].

The uniqueness and reliability values for the XRBR PUF are 40.67% and 98.22% respectively as shown in [13], which are very close to the ideal values. The comparison between the uniqueness and reliability values of BR PUF and XRBR PUF in [13] shows that XRBR PUF achieves better values. Also, XRBR PUF is a low-cost PUF design. As a memory-based PUF, the XRBR PUF is feasible,

and it can generate a comparably higher number of CRPs while reducing the hardware overheads. Now we proceed with the modelling attack.

3.2 Modelling XRBR PUF

The intuition behind the modeling of XRBR PUF will be of using an additive model for predicting the response. We have used an additive model where each XOR gate has weights that correspond to the difference between its pull-up strength and pull-down strength. To find the overall favoured response for a specific challenge, the weights are summed across all the XOR gates configured as an inverter. A higher preference for a positive response is indicated by a positive sum. The overall positive response is favoured by the pull-up strength of even positioned inverters and the pull-down strength of odd positioned inverters. Thus, the summation of weights requires negative and positive polarities.

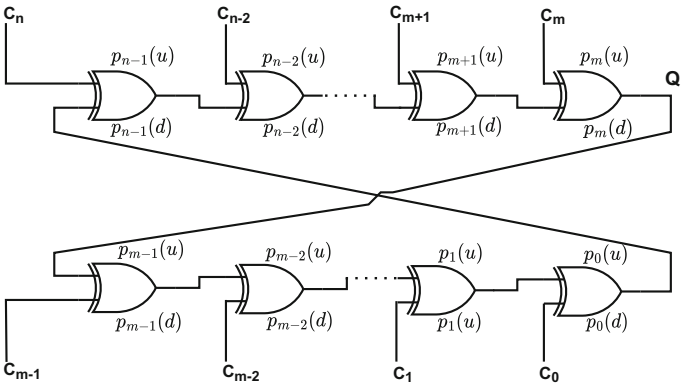


Fig. 8. XRBR PUF diagram showing the pull strengths of an XOR gate at each stage.

Let the the pull-up strength, pull-down strength and the difference between them for the i^{th} stage XOR gate be $p_i(u)$, $p_i(d)$ and p_i respectively (as shown in Fig. 8). On feeding the challenge (C) to the PUF circuit, the XOR gate of each stage will either be configured as a buffer or an inverter. The XOR gates that are configured as buffers will not contribute toward the response in any way. The XOR gates that are configured as inverters will contribute depending upon their position in the circuit. If an i^{th} stage XOR gate (configured as an inverter) is also an even positioned inverter in the PUF circuit, then it contributes towards the positive response with strength p_i, otherwise it contributes towards the positive response with strength $-p_i$.

Let us assume that the challenge bit for the i^{th} stage $C_i \in [-1, 1]$ is bipolar-encoded. Then we define the parity bit corresponding to the challenge bit as, $\Phi_i = -1^{s_i}.(1 + C_i)/2$. Here, s_i is the number of stages configured as inverters

till stage i. This provides the even or odd positioning of the stages that are configured as inverters.

The strength toward the positive response for any stage given a challenge bit C_i will be $\Delta S(i)$ and is given in Eq. 7,

$$\Delta S(i) = p_i.(-1^{s_i}\frac{1+C_i}{2}) \implies \Delta S(i) = p_i.\Phi_i \tag{7}$$

where the variable s_i is defined as:

$$s_i = \begin{cases} \frac{1+C_i}{2} & \textbf{if} \quad i = 0 \\ s_{i-1} + \frac{1+C_i}{2} & \textbf{if} \quad i = 1, ..., n-1 \end{cases}$$

The above mentioned notation not only nullifies the contribution of the stage configured as a buffer, but also lets the stage configured as an inverter contribute according to its position in the PUF circuit. For any challenge vector $C = \{C_0, C_1, ..., C_{n-1}\}$, the summed strengths toward the positive response is given in Eq. 8,

$$\begin{gathered} \Delta S = \sum_{i=0}^{n-1} \Delta S(i) \\ \Delta S = p_0.\Phi_0 + p_1.\Phi_1 + ... + p_{n-1}.\Phi_{n-1} \\ \Delta S = \mathbf{p}.\Phi^T \end{gathered} \tag{8}$$

where the feature vector Φ is defined as parity of challenge bits C:

$$\Phi_i = -1^{s_i}.\frac{1+C_i}{2}$$

The weight vector $\mathbf{p}$ is defined as follows:

$$\mathbf{p} = \{p_0, p_1, ..., p_{n-1}\}$$

According to our formulation, if the weight vector $\mathbf{p}$ (pull strengths) was known explicitly, then the sign of R_C could be used to predict the PUF response (Eq. 9).

$$R_C = sgn(\mathbf{p}.\Phi^T) \tag{9}$$

We will be using this additive model to simulate XRBR PUF and to evaluate it against SVM and LR algorithms in Sect. 4.

3.3 Mechanism of XRRO PUF

Next we proceed with the working principle of XRRO PUF. As discussed in Sect. 2.1, in CRO PUF, both the rings have stages consisting of two inverting delay elements and a multiplexer. The use of a inverter in a stage is dependent on the selection bit of the multiplexer of that stage. Now, to reduce the hardware overhead, XRRO PUF [13] employs single XOR gate at every stage in place

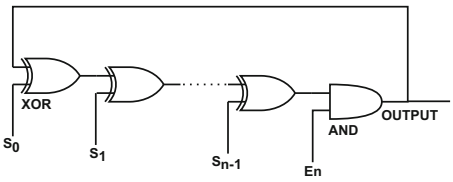


Fig. 9. The schematic diagram of an XRRO [13].

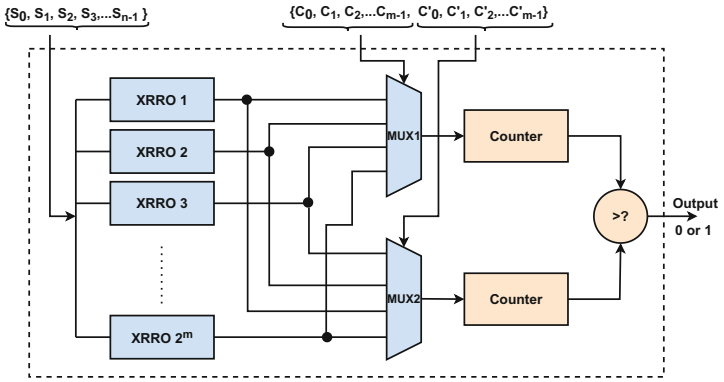


Fig. 10. The architecture of the XRRO PUF [13].

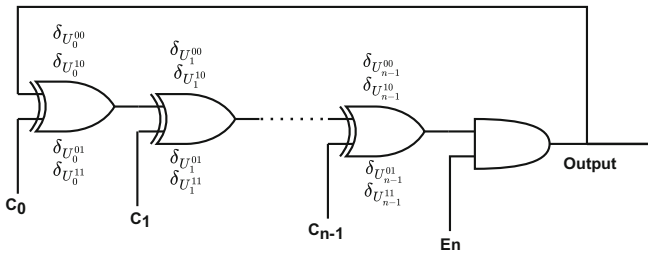
of the two inverters and a multiplexer. One input to the XOR gate acts as a configuration signal and the other input is connected to the output of the previous XOR gate, forming an XOR-based Ring Oscillator (as shown in Fig. 9).

Now, as shown in Fig. 1, a RO is generally made of odd number of logically inverting delay components. Thus the output of the last component is always logical "NOT" of the first input. Therefore the output of the RO oscillates between 0 and 1. Similarly for an XRRO, to oscillate, the number of inverting components should be odd. That is why, the configuration bits to all the XOR gates of an XRRO ring must contain an odd number of 1's so that odd number of XOR gates can be configured as inverters. The XRRO can construct up to $2^n/2$ different ROs from different configuration patterns, where n is the number of stages in the XRRO.

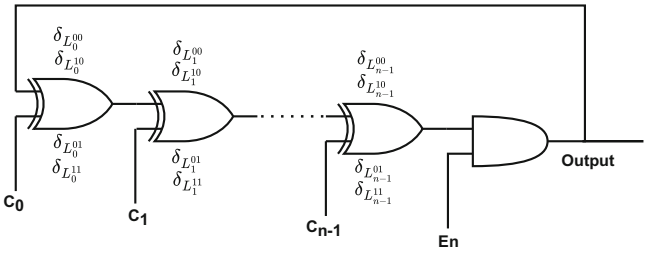
The architecture of the XRRO PUF is given in Fig. 10. We consider 2^m number of XRRO layers and all of them are configured using the same configuration signals $(S_0, S_1, ..., S_{n-1})$. The selection bits for both top $(C_0, C_1, ..., C_{m-1})$ and bottom $(C'_0, C'_1, ..., C'_{m-1})$ multiplexers chooses two different XRROs. The selected pair of XRROs only participate in the response generation process. The oscillation frequencies for both of them are compared based on which the response gets generated. If an XRRO selected by MUX1 has more oscillation frequency than the one selected by MUX2, response 1 is generated, otherwise response 0 is generated.

The uniqueness and reliability values for the XRRO PUF are 48.76% and 97.72% respectively as shown in [13], which are very close to the ideal values. The comparison between uniqueness and reliability values of RO PUF, CRO PUF and XRRO PUF in [13] shows that, XRRO PUF achieves better values.

3.4 Modelling XRRO PUF



(a) Delay components of upper XRRO.



(b) Delay components of lower XRRO.

Fig. 11. Delay components of the selected XRROs.

Next, we build up the mathematical model of the XRRO PUF. For upper and lower XRRO, there are two XOR gates. Every XOR gate contributes four delay elements corresponding to input values of '00', '01', '10' and '11'. So, every stage of XRRO PUF contains total eight delay elements based on the input values of the two XOR gates. Let the delay elements associated with stage i of the upper XRRO of be $\delta_{U_i^{00}}$, $\delta_{U_i^{01}}$, $\delta_{U_i^{10}}$ and $\delta_{U_i^{11}}$ for the inputs AB = "00", "01", "10" and "11" respectively (as shown in Fig. 11a). Similarly for stage i of the lower XRRO, the delays be, $\delta_{L_i^{00}}$, $\delta_{L_i^{01}}$, $\delta_{L_i^{10}}$ and $\delta_{L_i^{11}}$ (as shown in Fig. 11b). Consider the input C_i to the XOR gates be the configuration bit.

Now the main crux of our attack is to find out how the computation of the oscillation frequency of XRRO is different from RO/CRO PUF. In RO/CRO PUF, whatever be the input signal (0/1), it simply gets passed through the buffer depending on the multiplexer selection. But this is not the case in XRRO. Here two inputs of the XOR gate are involved at every stage. Now, selection bit is fixed for a particular stage, i.e. either 0/1. Now, depending on the output

of the previous stage, the inputs of the current stage can be either [00/10] or [01/11] and XOR logic is evaluated. And for every input pattern, the delay will be different. Thus two consecutive oscillation delays for the XRRO will be different due to the complement of the previous input. Finally, the frequency of the XRRO will be determined by the sum of the delay components that are selected by the configuration bits in two consecutive oscillations.

Hence, the delays selected for stage i for both upper and lower XRRO in two consecutive oscillations is given by the following equation where a challenge bit $C_i \in [-1, 1]$ is bipolar-encoded.

$$\begin{aligned} \Delta D_{upper}(i) &= \frac{1-C_i}{2}(\delta_{U_i^{00}} + \delta_{U_i^{10}}) + \frac{1+C_i}{2}(\delta_{U_i^{01}} + \delta_{U_i^{11}}) \\ \Delta D_{lower}(i) &= \frac{1-C_i}{2}(\delta_{L_i^{00}} + \delta_{L_i^{10}}) + \frac{1+C_i}{2}(\delta_{L_i^{01}} + \delta_{L_i^{11}}) \end{aligned} \tag{10}$$

The total delays across stages for both upper and lower XRROs will be (for two consecutive oscillations),

$$\begin{aligned} \Delta D_{upper} &= \sum_{i=0}^{n-1}\left[\frac{1-C_i}{2}(\delta_{U_i^{00}} + \delta_{U_i^{10}}) + \frac{1+C_i}{2}(\delta_{U_i^{01}} + \delta_{U_i^{11}})\right] \\ \Delta D_{lower} &= \sum_{i=0}^{n-1}\left[\frac{1-C_i}{2}(\delta_{L_i^{00}} + \delta_{L_i^{10}}) + \frac{1+C_i}{2}(\delta_{L_i^{01}} + \delta_{L_i^{11}})\right] \end{aligned} \tag{11}$$

The XRRO PUF response is generated by the comparison of ΔD_{upper} and ΔD_{lower}, generating a binary 1 if ΔD_{upper} has lesser value than ΔD_{lower} and 0 if it is the opposite. Let the difference be ΔD,

$$\begin{aligned} \Delta D = &\sum_{i=0}^{n-1}\left[\frac{1-C_i}{2}(\delta_{U_i^{00}} + \delta_{U_i^{10}}) + \frac{1+C_i}{2}(\delta_{U_i^{01}} + \delta_{U_i^{11}})\right] \\ &- \sum_{i=0}^{n-1}\left[\frac{1-C_i}{2}(\delta_{L_i^{00}} + \delta_{L_i^{10}}) + \frac{1+C_i}{2}(\delta_{L_i^{01}} + \delta_{L_i^{11}})\right] \end{aligned} \tag{12}$$

$$\begin{aligned} \Delta D = &\left[\sum_{i=0}^{n-1}\frac{1-C_i}{2}(\delta_{U_i^{00}} + \delta_{U_i^{10}}) - \sum_{i=0}^{n-1}\frac{1-C_i}{2}(\delta_{L_i^{00}} + \delta_{L_i^{10}})\right] \\ &+ \left[\sum_{i=0}^{n-1}\frac{1+C_i}{2}(\delta_{U_i^{01}} + \delta_{U_i^{11}}) - \sum_{i=0}^{n-1}\frac{1+C_i}{2}(\delta_{L_i^{01}} + \delta_{L_i^{11}})\right] \end{aligned} \tag{13}$$

$$\begin{aligned} \Delta D = &\left[\sum_{i=0}^{n-1}\frac{1-C_i}{2}(\delta_{U_i^{00}} + \delta_{U_i^{10}} - \delta_{L_i^{00}} - \delta_{L_i^{10}})\right] \\ &+ \left[\sum_{i=0}^{n-1}\frac{1+C_i}{2}(\delta_{U_i^{01}} + \delta_{U_i^{11}} - \delta_{L_i^{01}} - \delta_{L_i^{11}})\right] \end{aligned} \tag{14}$$

Let us assume,

$$\alpha_i = \delta_{U_i^{00}} + \delta_{U_i^{10}} - \delta_{L_i^{00}} - \delta_{L_i^{10}},\ \beta_i = \delta_{U_i^{01}} + \delta_{U_i^{11}} - \delta_{L_i^{01}} - \delta_{L_i^{11}}$$

Then,

$$\Delta D = \sum_{i=0}^{n-1} \frac{1 - C_i}{2}(\alpha_i) + \sum_{i=0}^{n-1} \frac{1 + C_i}{2}(\beta_i)$$

For convenience, we define $\overrightarrow{X}$ and $\overrightarrow{Y}$ such that,

$$\Delta D = \overrightarrow{X}\overrightarrow{W_\alpha} + \overrightarrow{Y}\overrightarrow{W_\beta} \tag{15}$$

where,

$$\overrightarrow{X} = \left\{\frac{1 - C_0}{2}, \frac{1 - C_1}{2}, ..., \frac{1 - C_{n-1}}{2}\right\},\ \overrightarrow{Y} = \left\{\frac{1 + C_0}{2}, \frac{1 + C_1}{2}, ..., \frac{1 + C_{n-1}}{2}\right\}$$

$$\overrightarrow{W_\alpha} = \left\{\alpha_0, \alpha_1, ..., \alpha_{n-1}\right\},\ \overrightarrow{W_\beta} = \left\{\beta_0, \beta_1, ..., \beta_{n-1}\right\}$$

In terms of a single weight vector and a single parity vector, the linear additive delay model of an XRRO PUF can be defined as,

$$\Delta D = \overrightarrow{Z}\overrightarrow{W} \tag{16}$$

where,

$$\overrightarrow{Z} = \left\{\frac{1 - C_0}{2}, \frac{1 - C_1}{2}, ..., \frac{1 - C_{n-1}}{2}, \frac{1 + C_0}{2}, \frac{1 + C_1}{2}, ..., \frac{1 + C_{n-1}}{2}\right\}$$

$$\overrightarrow{W} = \left\{\alpha_0, \alpha_1, ..., \alpha_{n-1}, \beta_0, \beta_1, ..., \beta_{n-1}\right\}$$

The frequency for the upper and lower XRRO selected of the XRRO PUF is given to be $F_{upper} = 2/\Delta D_{upper}$ and $F_{lower} = 2/\Delta D_{lower}$ respectively. If upper XRRO has a greater frequency then the binary response will be 1, else it will be 0. Thus, the response depends on the difference in the total delays for the two consecutive oscillations of both the XRROs(ΔD). This model has been used in Sect. 4 below to simulate an XRRO PUF and to evaluate it against the SVM and LR based attacks. Please note that we have to follow the same procedure as mentioned above for every pair of XRROs and create an instance of SVM/LR model. The collaborative accuracy of all such models provide us the ultimate accuracy of the XRRO PUF instance.

4 Machine Learning Attacks on XOR-Based Reconfigurable PUFs

Now we present our experimental set up and results on the prediction accuracy of modelling attacks on XRRO and XRBR PUFs.

Table 1. SVM attack on XRBR PUF.

Training CRPs	32 bit XRBR PUF	64 bit XRBR PUF	128 bit XRBR PUF
1000	92.40%	91.40%	90.20%
3000	95.76%	94.96%	93.76%
5000	96.32%	95.86%	93.94%
8000	96.60%	96.68%	95.36%
10000	97.55%	96.27%	95.73%

Table 2. SVM attack on XRRO PUF with 8 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
10000	87.21%	83.94%	75.21%
40000	91.65%	90.11%	86.97%
70000	94.25%	93.88%	89.69%
100000	95.32%	94.17%	91.38%
120000	96.18%	94.67%	93.16%
150000	96.14%	94.97%	92.81%
200000	96.57%	95.41%	94.00%

Table 3. SVM attack on XRRO PUF with 16 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
50000	86.33%	83.18%	78.88%
100000	90.55%	87.12%	81.98%
200000	93.96%	90.88%	87.49%
400000	93.90%	94.37%	90.71%
600000	95.76%	95.24%	92.95%
800000	96.27%	95.52%	93.59%
1000000	96.59%	95.61%	94.31%

SVM Attacks: A SVM classifies data by finding the best hyperplane that separates all data points belonging to one class from those belonging to the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes. The SVM implementation in *scikit-learn* library is used for this work which can be found here [2].

Using the mathematical model discussed in Sect. 3.2, the SVM attack is performed on the XRBR PUF by training the model on the given CRP samples. The feature vector Φ is determined given the challenge vector (C) in the CRP, and the model is trained on that feature vectors and the associated responses. The response of the XRBR PUF(R_C) for the challenge vector C was formulated as $R_C = sgn(\mathbf{p}.\Phi^T)$ where $\mathbf{p}$ was defined as $\{p_0, p_1, ..., p_{n-1}\}$. In SVM formulation, the p_i terms do not appear explicitly as the classifier simply works to

find the hyperplane with the largest margin to separate the challenges into two classes based on their responses. Table 1 shows the prediction rates for a 32, 64, and 128 bit XRBR PUF using the SVM method with training sample sizes (CRPs) ranging from 1000 to 10,000. Given 10,000 training CRPs, it is possible to predict the XRBR PUF design with greater than 95% accuracy, even for a large 128 bit XRBR PUF.

Table 4. SVM attack on XRRO PUF with 32 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
500000	91.83%	88.23%	84.60%
1000000	94.35%	92.12%	88.93%
1500000	94.35%	93.64%	90.45%
2000000	95.39%	94.23%	92.26%
3000000	96.27%	95.27%	93.61%

Table 5. SVM attack on XRRO PUF with 64 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
1000000	88.24%	84.78%	79.41%
1500000	90.27%	87.05%	81.84%
2000000	91.41%	88.50%	84.01%
3000000	92.93%	90.49%	86.98%
5000000	95.02%	92.60%	89.82%
8000000	96.24%	94.239%	92.14%

Table 6. LR attack on XRBR PUF.

Training CRPs	32 bit XRBR PUF	64 bit XRBR PUF	128 bit XRBR PUF
1000	97.10%	96.40%	94.30%
3000	98.80%	98.50%	98.40%
5000	99.22%	98.76%	98.30%
8000	99.12%	98.95%	98.97%
10000	99.53%	99.40%	99.06%

For attacking the XRRO PUF, We have created an instance of SVM model for each pair of XRROs. For each challenge provided, we separate the selection bits of both the multiplexers and the configurable bits to the XRROs. Identifying the pair of XRROs selected by the selection bits (of both multiplexers), we train the corresponding SVM model to the pair of XRROs selected. The training is been done on the configuration bits of the XRROs (i.e., C as going

by the notation discussed in Sect. 3.4). The feature vector $\overrightarrow{Z}$ is derived from the vector C and the model is trained with the feature vectors and the associated responses. In the SVM formulation, no weight terms α_i or β_i (please ref to Sect. 3.4) appear explicitly as the classifier simply works to find the hyperplane with the largest margin to separate the challenges (Here challenges refers to the configuration signals to the XRROs) into two classes based on their responses. For the experimental purpose, we have chosen 8, 16, 32 and 64 layer XRRO PUF where every XRRO is of 31, 63 and 127 stage long. Table 2, 3, 4, 5 shows the prediction rates for a 31, 63, and 127 stage XRRO PUF having 8, 16, 32 and 64 layers respectively.

LR Attacks: LR method builds a linear model of the system using the correlation between an independent and dependent variable of a known training set. The LR implementation in *scikit-learn* library is used for this work [1].

Table 7. LR attack on XRRO PUF with 8 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
10000	90.96%	88.75%	82.65%
40000	97.31%	95.58%	93.48%
70000	98.03%	97.20%	95.29%
100000	98.53%	97.81%	96.99%
120000	98.71%	98.13%	97.17%
150000	98.94%	98.56%	97.42%
200000	99.09%	98.79%	98.30%

Table 8. LR attack on XRRO PUF with 16 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
50000	93.29%	90.64%	85.03%
100000	95.18%	93.36%	90.39%
200000	97.21%	95.87%	93.76%
400000	98.56%	97.81%	96.79%
600000	98.89%	98.37%	97.56%
800000	99.10%	98.71%	97.72%
1000000	99.21%	98.85%	98.29%

Table 9. LR attack on XRRO PUF with 32 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
500000	95.97%	94.25%	91.45%
1000000	96.94%	96.78%	94.95%
1500000	98.36%	97.61%	96.12%
2000000	98.65%	98.04%	97.00%
3000000	99.11%	98.59%	97.76%

Table 10. LR attack on XRRO PUF with 64 layers of XRROs.

Training CRPs	31 stage XRRO PUF	63 stage XRRO PUF	127 stage XRRO PUF
1000000	93.49%	91.02%	86.29%
1500000	95.64%	92.95%	88.90%
2000000	96.29%	94.22%	90.96%
3000000	97.04%	95.85%	93.34%
5000000	98.21%	96.97%	95.89%
8000000	99.02%	98.06%	97.07%

For attacking the XRBR PUF, The training is done on the CRPs set after transforming the challenge bits as discussed in Sect. 3.2. The model is then trained on the transformed CRPs set. Table 6 shows the prediction rates for a 32, 64, and 128 bit XRBR PUF using the LR method with training sample sizes(CRPs) ranging from 1000 to 10,000. Given 10,000 training CRPs, it is possible to predict the XRBR PUF design with greater than 99% accuracy, even for a large 128 bit XRBR PUF.

Similarly, using the modelling of XRRO PUF discussed in Sect. 3.4, we have performed the LR attack on XRRO PUF. Table 7, 8, 9, 10 shows the prediction rates for a 31, 63, and 127 stage XRRO PUF having 8, 16, 32 and 64 layers respectively. Finally we can conclude from the above mentioned results that LR method provides better prediction accuracy than SVM method in the case of both XRBR and XRRO PUF.

5 Conclusion

In this work, we investigate the security metrics of XRBR and XRRO PUFs which are recently proposed for generating hardware fingerprints in the resource constrained devices for IoT frameworks. Though the PUF architectures demand very low hardware overhead by maintaining substantial uniformity and reliability properties, the strengths of these designs against mathematical modeling was yet unexplored. To the best of our knowledge, this is the first work that tries to make predictive models for the same and scrutinises the vulnerabilities against machine learning attacks. We leverage a common flaw of not incorporating any non-linear elements in the designs and show how that makes both schemes prone to ML attacks. Hence these designs are not any better than a simple RO PUF or BR PUF design. Finally with the experimental validation we have shown that both the designs can be broken using SVM and LR algorithms with the accuracy of approximately upto 99%. Overall, reducing the hardware overhead of such architectures without being prone to ML attacks could be a very challenging research area and can be a potential direction for future work.

References

1. Scikit-learn Logistic Regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
2. Scikit-learn Support Vector Machine. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
3. Becker, G.T., Kumar, R.: Active and passive side-channel attacks on delay based puf designs. Cryptology ePrint Archive (2014)
4. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: The bistable ring PUF: a new architecture for strong physical unclonable functions. In: 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 134–141. IEEE (2011)
5. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: Characterization of the bistable ring PUF. In: 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1459–1462. IEEE (2012)
6. Gassend, B., Clarke, D., Van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 148–160 (2002)
7. Gu, C., Hanley, N., O'neill, M.: Improved reliability of FPGA-based PUF identification generator design. ACM Trans. Reconfigurable Technol. Syst. (TRETS). **10**(3), 1–23 (2017)
8. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_5
9. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: Physical unclonable functions and public-key crypto for FPGA ip protection. In: 2007 International Conference on Field Programmable Logic and Applications, pp. 189–195. IEEE (2007)
10. Holcomb, D.E., Fu, K.: Bitline PUF: building native challenge-response PUF capability into any SRAM. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 510–526. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_28
11. Hospodar, G., Maes, R., Verbauwhede, I.: Machine learning attacks on 65nm arbiter PUFs: accurate modeling poses strict bounds on usability. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 37–42. IEEE (2012)
12. Lim, D.: Extracting secret keys from integrated circuits in master thesis. Massachusetts Institute of Technology (2004)
13. Liu, W., et al.: XOR-based low-cost reconfigurable PUFs for IoT security. ACM Trans. Embed. Comput. Syst. (TECS) **18**(3), 1–21 (2019)
14. Lofstrom, K., Daasch, W.R., Taylor, D.: IC identification circuit using device mismatch. In: 2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056), pp. 372–373. IEEE (2000)
15. Maes, R., Rozic, V., Verbauwhede, I., Koeberl, P., Van der Sluis, E., van der Leest, V.: Experimental evaluation of physically unclonable functions in 65 nm CMOS. In: 2012 Proceedings of the ESSCIRC (ESSCIRC), pp. 486–489. IEEE (2012)
16. Maiti, A., Schaumont, P.: Improved ring oscillator PUF: an FPGA-friendly secure primitive. J. Cryptology **24**(2), 375–397 (2011)
17. Miskelly, J., Gu, C., Ma, Q., Cui, Y., Liu, W., O'Neill, M.: Modelling attack analysis of configurable ring oscillator (CRO) PUF designs. In: 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), pp. 1–5. IEEE (2018)

18. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**(5589), 2026–2030 (2002)
19. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 237–249 (2010)
20. Rührmair, U., et al.: PUF modeling attacks on simulated and silicon data. IEEE Trans. Inf. Forensics Secur. **8**(11), 1876–1891 (2013)
21. Sahoo, D.P., Nguyen, P.H., Chakraborty, R.S., Mukhopadhyay, D.: Architectural bias: a novel statistical metric to evaluate arbiter PUF variants. IACR Cryptol. ePrint Arch. **2016**, 57 (2016)
22. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: 2007 44th ACM/IEEE Design Automation Conference, pp. 9–14. IEEE (2007)
23. Tehranipoor, F., Karimian, N., Yan, W., Chandy, J.A.: Dram-based intrinsic physically unclonable functions for system-level security and authentication. IEEE Trans. Very Large Scale Integr. Syst. **25**(3), 1085–1097 (2016)
24. Xu, X., Rührmair, U., Holcomb, D.E., Burleson, W.: Security evaluation and enhancement of bistable ring PUFs. In: Mangard, S., Schaumont, P. (eds.) RFIDSec 2015. LNCS, vol. 9440, pp. 3–16. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24837-0_1
25. Zhou, C., Parhi, K.K., Kim, C.H.: Secure and reliable XOR arbiter PUF design: an experimental study based on 1 trillion challenge response pair measurements. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2017). https://doi.org/10.1145/3061639.3062315