MXPUF: Secure PUF Design against State-of-the-art Modeling Attacks

Phuong Ha Nguyen[1], Durga Prasad Sahoo[2], Chenglu Jin[1], Kaleel Mahmood[1], and Marten van Dijk[1]

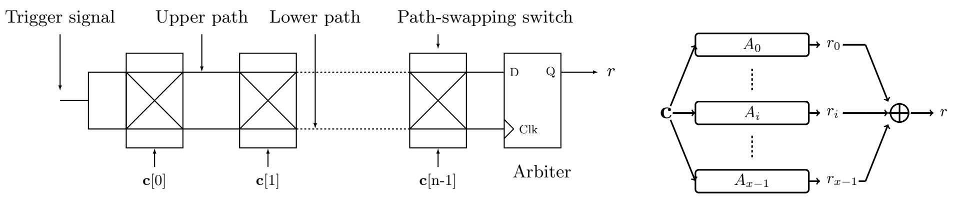
[1] University of Connecticut, USA,
[2] Robert Bosch India (RBEI/ETI)

Abstract. Silicon Physical Unclonable Functions (PUFs) have been proposed as an emerging hardware security primitive in various security applications such as device identification, authentication, and cryptographic key generation. Current so-called 'strong' PUFs, which allow a large challenge response space, are compositions of Arbiter PUFs (APUFs), e.g. the x-XOR APUF. Wide scale deployment of state-of-the-art compositions of APUFs, however, has stagnated due to various mathematical and physical attacks leading to software models that break the unclonability property of PUFs. The current state-of-the-art attack by Becker, CHES 2015, shows that the XOR APUF can be broken by modeling its APUF components separately thanks to CMA-ES, a machine learning algorithm, based on reliability information of measured XOR APUF responses. Thus, it is an important problem to design a strong PUF which can resist not only traditional modeling attacks but also Becker's attack. In this paper, we propose a new strong PUF design called (x, y)-MXPUF, which consists of two layers; the upper layer is an n-bit x-XOR APUF, and the lower layer is an $(n+1)$-bit y-XOR APUF. The response of x-XOR APUF for an n-bit challenge $\mathbf{c}$ in the upper layer is inserted at the middle of $\mathbf{c}$ to construct a new $(n+1)$-bit challenge for the y-XOR APUF in the lower layer giving the final response bit of the (x, y)-MXPUF. The reliability of (x, y)-MXPUF can be theoretically and experimentally shown to be twice the reliability of $(x + y)$-XOR PUF. In the context of traditional modeling attacks, when we keep the same hardware size, the security of (x, y)-MXPUF is only slightly weaker than that of $(x + y)$-XOR PUF. Our main contribution proves that the (x, y)-MXPUF is secure against Becker's attack.

Keywords: Arbiter physically unclonable function (APUF), majority voting, modeling attack, propagation criterion, reliability based modeling, XOR APUF.

1 Introduction

Intuitively, a silicon Physical Unclonable Function (PUF) [10] is a fingerprint of a chip which behaves as a one-way function in the following way: It offers *Manufacturing Resistance* as it leverages process manufacturing variation to generate



(a) n-bit APUF (b) x-XOR APUF

Fig. 1: Arbiter PUF (APUF) and XOR APUF. A x-XOR APUF consists of x APUFs $A_0, \ldots, A_{x-1}$.

a unique function taking "challenges" as input and generating "responses" as output; The function is *HW Unclonable* in that it cannot be cloned in hardware (the PUF's internal behavior, e.g. its unique physical characteristics or behavior of its wires, cannot be read out accurately enough; also it is not feasible to manufacture two PUFs with the same responses to a significant subset of challenges) and is *SW Unclonable* in that it cannot be efficiently learned given a "polynomial number" of challenge response pairs (making it impossible to impersonate/clone the function's behavior to a new random challenge in software).

The general concept of a PUF was first introduced by Pappu [19]. Silicon PUFs have been proposed as central building blocks for device identification and authentication [15], binding software to hardware platforms [11], secure storage of cryptographic secrets [29], and secure protocol design [3].

Fig. 1a depicts an Arbiter PUF (APUF) [10,13,14] with challenge input vector $\mathbf{c} \in \{0,1\}^n$ and response bit $r \in \{0,1\}$. An APUF is a strong PUF due to the large (2^n) number of Challenge Response Pairs (CRPs) which is infeasible to enumerate. Fig. 1b depicts a XOR APUF [24] which XORs the response bits of component APUFs into a final output response bit. In PUF literature, APUF variants and Lightweight Secure PUFs [16] are called lightweight PUFs because of their small hardware footprint. It has been shown that all these PUF constructions are vulnerable to cryptanalysis attacks or purely mathematical machine learning based modeling attacks [20,22,27,8,9,7]. In these attacks, a mathematical model of a PUF instance is built based on PUF inputs and outputs (CRPs) or some side channel information by using machine learning techniques such as Logistic Regression (LR), Support Vector Machine (SVM), Evolution Strategy (ES), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) etc [4,6,21,26,2,25].

In CHES 2015 Becker presented for the first time an efficient reliability-based modeling attack on x-XOR APUFs and another variant called x-way XOR APUFs, which uses CMA-ES to optimize its mathematical model [2]. This attack is based on the following observation: the repeatability or short-term reliability and the linear delay model of a given APUF instance (or the mathematical model of APUF) are strongly correlated [4,2]. Thus, instead of using CRPs directly for building component APUF models, each challenge is given as input to the PUF

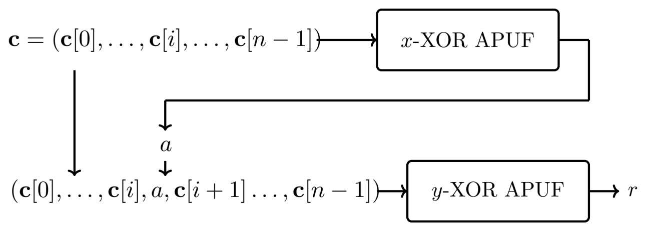

$\mathbf{c} = (\mathbf{c}[0], \ldots, \mathbf{c}[i], \ldots, \mathbf{c}[n-1])$
x-XOR APUF
a
$(\mathbf{c}[0], \ldots, \mathbf{c}[i], a, \mathbf{c}[i+1] \ldots, \mathbf{c}[n-1])$
y-XOR APUF
r

Fig. 2: (x, y)-MXPUF (Multiple XOR APUF) comprises of x-XOR APUF and y-XOR APUF.

multiple times and the resulting repeatability information of its response is used in the attack. Due to the structure of XOR APUF (see Fig. 1b), the reliability information of each individual component APUF *equally* affects the output of the XOR APUF. Becker showed that if we apply the CMA-ES learning algorithm on XOR APUF many times, then models for all the component APUFs can be built: Since each component APUF equally affects the output of the XOR APUF, the reliability information of a random batch of challenges will be most correlated to one of the component APUFs, the component which happens to contribute the most to the reliability information. This allows a divide-and-conquer attack which builds each component APUF model separately.

In this paper we first perform analysis together with simulations to understand the circumstances under which Becker's attack is successful. Next we propose a new PUF design, coined the Multiple XOR PUF (MXPUF), see Fig. 2. A (x, y)-MXPUF consists of two layers; the upper layer is an n-bit x-XOR APUF, and the lower layer is an $(n + 1)$-bit y-XOR APUF. The response of x-XOR APUF for an n-bit challenge $\mathbf{c}$ in the upper layer is inserted at the middle of $\mathbf{c}$ to construct a new $(n + 1)$-bit challenge for the y-XOR APUF in the lower layer giving the final response bit of the (x, y)-MXPUF. The reliability of (x, y)-MXPUF can be theoretically and experimentally shown to be twice of reliability of $(x + y)$-XOR APUF. Moreover, when keeping the same hardware footprint, the security of our (x, y)-MXPUF is slightly weaker than that of $(x + y)$-XOR APUF in context of traditional modeling attacks [20,22,27]. Our main contribution proves that the (x, y)-MXPUF is secure against Becker's attack.

Contributions: Our contributions are as follows:

1. We revisit Becker's attack on APUF variants in [2], and present two new metrics to enhance Becker's attack which require fewer challenge response pairs compared to the original one.
2. We analyze in detail Becker's attack on the x-XOR APUF:
   We conclude that if, compared to other APUF components, an APUF component contributes significantly less to the reliability information at the output (final response) of an XOR APUF, then its model *cannot* be built using the CMA-ES algorithm and Becker's attack fails.

3. We propose a new lightweight PUF design called (x, y)-MXPUF which has the following properties:
   (a) Compared to the $(x + y)$-XOR APUF (which has the same number of multiplex units as the (x, y)-MXPUF) it is less prone to measurement noise if we carefully place the feedback position a in Fig. 2 based on the Propagation Criterion (PC) property [17,5,18].
   (b) It is secure against Becker's attack [2] because, compared to the lower layer y-XOR APUF, the upper layer x-XOR APUF affects the reliability information of the final response bit much less. In addition, we argue that it is secure against a more powerful attack which combines Becker's attack [2] with fault analysis in [6]. We show that (x, y)-MXPUF is only slightly weaker than $(x+y)$-XOR APUF in terms of security with respect to previously known classical machine learning attacks.
4. Finally, we verify our observations using simulated results, where APUF instances are simulated in Matlab. The simulation results show that reliability of (x, y)-MXPUF is twice the reliability of $(x + y)$-XOR APUF.

Organization: The rest of the paper is organized as follows. In Section 2, the background of Arbiter PUFs is briefly provided. Section 3 describes Becker's attack on x-XOR APUF after which we detail experiments in order to understand under what circumstances Becker's attack cannot be successful. Based on the knowledge gained in Section 3, we develop our MXPUF design and provide in Section 4 its security analysis and reliability property. Section 5 presents our experimental results and Section 6 concludes the paper. Due to page limitation, we discuss how to improve the efficiency of the attack in [2] in Appendix A and B.

2 Background Arbiter PUF

In this section, we briefly introduce the analytical delay model [13], and reliability model [4] of APUF; this is needed to understand Becker's attack and the newly proposed (x, y)-MXPUF.

2.1 Linear Additive Delay Model

In [13,14], an analytical model which is called *Linear Additive Delay Model* is presented. As shown in [13], the linear additive delay model of an APUF has the form:

$$\Delta = \mathbf{w}[0]\mathbf{\Phi}[0] + \cdots + \mathbf{w}[i]\mathbf{\Phi}[i] + \cdots + \mathbf{w}[n]\mathbf{\Phi}[n] = \mathbf{w}^{\mathsf{T}}\mathbf{\Phi}, \tag{1}$$

where $\mathbf{w}$ and $\mathbf{\Phi}$ are known as *weight and parity (or feature) vectors*, respectively. The parity vector $\mathbf{\Phi}$ is derived from the challenge $\mathbf{c}$ as follows:

$$\mathbf{\Phi}[n] = 1, \quad \text{and} \quad \mathbf{\Phi}[i] = \prod_{j=i}^{n-1}(1 - 2\mathbf{c}[j]), i = 0, \ldots, n-1. \tag{2}$$

In this delay model, the unknown weight vector $\mathbf{w}$ depends on the process variation of the PUF instance (i.e. of a specifically manufactured PUF). The response to a challenge $\mathbf{c}$ is defined as: If $\Delta \geq 0$, then $r = 0$. Otherwise, $r = 1$.

In order to break the SW unclonability property of an APUF, an adversary can build an accurate mathematical model using machine learning techniques (e.g. Support Vector Machine, Logistic Regression, Evolution Strategy) based on this linear model [13,23,20] and a recorded set of challenge-response pairs (CRPs). Because APUFs are vulnerable to machine learning based modeling attacks, the XOR Arbiter PUF (XOR APUF) was introduced in [24] (See Fig 1b). Due to the non-linearity of the PUF models introduced by XOR operation, traditional modeling attacks can be prevented (i.e. can become impractical) if a sufficient number of component APUFs are employed in the XOR APUF [20,27].

2.2 Repeatability Models of APUFs

In [4], Delvaux et al. introduced the first modeling attack on APUFs which exploits the reliability (or repeatability) information of an APUF response. To make the paper self-contained, we briefly introduce the repeatability model of an APUF and the key idea of the attacks presented in [4].

Variability vs. Noise Typically, manufacturing variability and measurement noise are two undesired phenomenons in an electric circuit [4]. Variability is caused by the manufacturing processes, and noise is a random temporal phenomenon, for example instability of the supply voltage or change of circuit temperature. While noise is a non-reproducible physical feature, variability is reproducible and is exploited to build PUFs. Due to noise, the reproducibility or reliability of PUF output is not perfect, i.e., less than 100%.

Repeatability Model of an APUF The repeatability is the *short-term reliability of a PUF* in presence of CMOS noise, and it is not the long-term device ageing effect [4]. For a given challenge $\mathbf{c}$, a repeatability R is measured as follows. Assume that the challenge $\mathbf{c}$ is evaluated M times, and suppose that the measured responses are equal to 1 in N out of M evaluations. The repeatability is defined as $R = N/M \in [0, 1]$. As shown in [4], the relationship between the delay difference Δ and $R \in [0.1, 0.9]$ of a given challenge $\mathbf{c}$ can be described as:

$$\Delta/\sigma_N = \sum_{i=0}^{n} (\mathbf{w}[i]/\sigma_N)\mathbf{\Phi}[i] = -\Phi^{-1}(R) \approx R, \tag{3}$$

where the noise follows a normal distribution $\mathcal{N}(0, \sigma_N)$ and $\Phi(\cdot)$ is the probability distribution function of the standard normal distribution.

Due to noise, the repeatability R of a given challenge $\mathbf{c}$ leaks information about weights $\mathbf{w}[0], \ldots, \mathbf{w}[n]$ in the analytical expression of Δ. Conceptually, instead of using CRPs in traditional modeling attacks, the *repeatability* of APUF outputs can be used to build an APUF model. Based on a set of challenge reliability (not response) pairs $(\mathbf{c}, R)$ with $R \in [0.1, 0.9]$, a system of linear equations is established and the weights $\mathbf{w}[i]/\sigma_N$ can be solved by using the *Least Mean Square* algorithm [4]. This can be used to predict the response r to a challenge $\mathbf{c}$ based on the comparison $\Delta/\sigma_N \approx R \leq$ or ≥ 0. Hence, we can follow traditional modeling attacks based on the linear additive delay model, which use machine learning techniques and a measured set of CRPs (now replaced by a set of $(\mathbf{c}, R)$ pairs) to built a model for the APUF.

In general, $R = \Phi(-\Delta/\sigma_N)$, and this formula is exploited for enhancing Becker's attack [2] in Appendix A.

2.3 Repeatability based CMA-ES on APUFs

In [1,2] Becker develops a modeling attack on APUFs using CMA-ES with reliability information obtained from the repeated measurements of CRPs. More precisely, reliability information R of a challenge $\mathbf{c}$ (i.e. $(\mathbf{c}, R)$) is used in the attack instead of the corresponding response r (i.e. $(\mathbf{c}, r)$).

The rationale behind this attack is as follows: if the delay difference $|\Delta|$ between the two competing paths in an APUF for a given challenge $\mathbf{c}$ is smaller than a threshold ϵ, then the corresponding response r would be unreliable in the presence of noise; otherwise the response would be reliable. This implies that reliability information directly leaks information about the 'wire delays' in an APUF model. Unlike CRP-based modeling attacks [20] where noisy (or unreliable) CRPs reduce the modeling accuracy, the unreliable CRPs are exploited in CMA-ES based modeling.

Let r_i be the i-th measured response of challenge $\mathbf{c}$ for $i = 1, \ldots, M$. Two different definitions of R, as provided in Eq. (4) and Eq. (5), are found in [4] and [2], respectively:

$$R = \frac{1}{M} \sum_{i=1}^{M} r_i \tag{4}$$

$$R = |M/2 - \sum_{i=1}^{M} r_i| \tag{5}$$

Let us recall the CMA-ES based modeling attack on an APUF as discussed in [1,2]. The objective of CMA-ES is to learn weights $\mathbf{w} = (\mathbf{w}[0], \ldots, \mathbf{w}[n])$ together with a threshold value ϵ. All variables $\mathbf{w}[0], \ldots, \mathbf{w}[n-1]$ are treated as independent and identically distributed Gaussian random variables. The attack is conducted as follows:

1. Collect N challenge-reliability pairs $\mathcal{Q} = \{(\mathbf{c}_1, R_1), \ldots, (\mathbf{c}_i, R_i), \ldots, (\mathbf{c}_N, R_N)\}$.
2. Generate K random models: $\{(\mathbf{w}_1, \epsilon_1), \ldots, (\mathbf{w}_j, \epsilon_j), \ldots, (\mathbf{w}_K, \epsilon_K)\}$.
3. For each model $(\mathbf{w}_j, \epsilon_j)$ $(j = 1, \ldots, K)$, do the following steps:
   (a) For each challenge $\mathbf{c}_i$ $(i = 0, \ldots, N)$, compute the R'_i as follows:

$$R'_i = \begin{cases} 1, & \text{if } |\Delta| \geq \epsilon \\ 0, & \text{if } |\Delta| < \epsilon, \end{cases} \tag{6}$$

where $\epsilon = \epsilon_j$ and Δ follows from (1) and (2) with $\mathbf{c} = \mathbf{c}_i$ and $\mathbf{w} = \mathbf{w}_j$.

(b) Compute the Pearson correlation ρ_j based on the $(R_1, \dots, R_i, \dots, R_N)$ and $(R'_1, \dots, R'_i, \dots, R'_N)$, i.e.

$$\rho_j = \frac{\sum_{i=1}^{N}(R_i - \bar{R}) \times (R'_i - \bar{R}')}{\sqrt{\sum_{i=1}^{N}(R_i - \bar{R})^2} \times \sqrt{\sum_{i=1}^{N}(R'_i - \bar{R}')^2}},$$

where $\bar{R} = \frac{\sum_{i=1}^{N} R_i}{N}$ and $\bar{R}' = \frac{\sum_{i=1}^{N} R'_i}{N}$.

4. CMA-ES keeps L models $(\mathbf{w}_j, \epsilon_j)$ which have the highest Pearson correlation ρ, and then, from these L models, another K models are generated based on the CMA-ES algorithm.
5. Repeat the steps (3)-(4) for H iterations and model $(\mathbf{w}, \epsilon)$ which has highest Pearson correlation ρ will be chosen as the correct model. It is noted that sometimes the chosen model can have a poor prediction accuracy and in this case we restart the algorithm to find a model with higher prediction accuracy.

Compared to traditional modeling attacks, the CMA-ES attack in [2] has the following main advantage: There is no *noisy information or wrong information* in the training dataset. In traditional modeling attacks, noisy CRPs are treated as noise, i.e., they are unwanted information (or wrong information) in training data. This fact strongly affects the prediction accuracy of the constructed model. In contradiction to traditional modeling attacks, the CMA-ES attack in [2] does not have *wrong information* in the training dataset. Particularly, noisy CRPs are the most valuable information for constructing models in this approach.

3 Becker's Attack on XOR APUF: Insights

An x-XOR APUF consists of x n-bit APUF instances $A_0, \dots, A_{x-1}$ as depicted in Fig. 1b. For a given challenge $\mathbf{c}$, the response r of an x-XOR APUF is generated as follows:

$$r = r_0 \oplus \cdots \oplus r_i \oplus \cdots \oplus r_{x-1}, \tag{7}$$

where $r_i = A_i(\mathbf{c}), i = 0, \dots, x-1$. According to traditional ML-based modeling, the modeling complexity of an x-XOR APUF increases exponentially in x. However, Becker's attack [2] on XOR APUF achieves a modeling complexity that increases linearly in x. In this section, we explain insights into Becker's attack as understanding the rationale behind this attack helps us to develop our strong MXPUF design that is secure against Becker's attack.

Let $\mathcal{Q} = \{(\mathbf{c}_1, R_1), \dots, (\mathbf{c}_i, R_i), \dots, (\mathbf{c}_N, R_N)\}$ be a set of challenge and reliability information pairs of an x-XOR APUF instance. In [2] Becker reported an important observation in the context of x-XOR APUF modeling: if the CMA-ES algorithm for modeling an APUF (see Section 2.3) is executed many times with the set $\mathcal{Q}$, then it can produce x different models for $A_0, \dots, A_{x-1}$ with high probability. Although there is no proof on how many times the CMA-ES algorithm has to be executed to get the models of all APUF instances of the x-XOR APUF, experimentally it is observed that this value needs not to be large. As done in [2] one can parallelize CMA-ES executions to built models for $A_0, \ldots, A_{x-1}$. Thus the modeling of x-XOR APUF boils down to the modeling of x independent APUF instances.

Now, we consider the following questions for in-depth analysis of Becker's attack:

1. As Becker mentioned in [2] as a fact, in practice some APUFs are *easier* and some are *harder* to model using the given challenges and reliability information pairs. It turns out that CMA-ES converges more often to some PUF instances than others. Why does this happen?
2. In [2], Becker did not clearly mention in different runs *whether* the same set $\mathcal{Q}$ is used or not. Thus we wonder whether Becker's attack can build the models of *all* APUF instances in XOR APUF if the same data set $\mathcal{Q}$ is used in different runs.
3. What are the conditions such that CMA-ES never converges to a particular APUF instance? In other words, under which condition does the attack fail?

In [2] Becker posed question-1 without any theoretical answer and question-3 is still not investigated in the literature. The next experiments are aimed at answering these questions; based on the knowledge gained from these experiments, we develop an APUF composition that is secure against Becker's reliability-based modeling attack as well as traditional mathematical modeling attacks.

3.1 Experiment-I

Objective of this experiment is to investigate why some APUF instances are *easier* and some are *harder* to model by using reliability-based CMA-ES. More specifically, we want to verify whether the probability of being converged to in CMA-ES is correlated with the noise rate of each APUF instance presenting *in a given data set* $\mathcal{Q}$, where the noise rate is the number of noisy CRPs divided by the number of total CRPs in $\mathcal{Q}$.

Setup. We perform the attack on 10-XOR APUF by executing CMA-ES algorithm for APUF 100 times on different data sets $\mathcal{Q}$. The 10-XOR APUF is simulated using Matlab as described in Section 5, and the noise rate of all APUFs are set at 20%. Notice that, the noise rate is set to be 20%, but for a given data set $\mathcal{Q}$, the noise rate of each APUF instance presents in $\mathcal{Q}$ is not guaranteed to be 20%. In each run, we randomly generate set $\mathcal{Q}$ of size 70×10^3, and each challenge is evaluated 11 times. Note that, in this experiment, we know the configuration of each PUF instances, i.e., we are not a *true* attacker who does not have this knowledge.

Result and Discussion. In each run of CMA-ES, an APUF model $\mathbf{w}$ is produced. Since we know the configurations and CRPs of each individual APUF in

Table 1: Correlation between the noise rate ranking presenting in each $\mathcal{Q}$ and probability of correct models of APUF instances over 100 runs of CMA-ES

Rank	0	1	2	3	4	5	6	7	8	9	10
Count	2	36	5	9	5	12	7	7	5	7	5

XOR APUF, we do the following classification of the built model: if the model $\mathbf{w}$ matches with any of the APUF instances with probability ≥ 0.9 or ≤ 0.1 (complemented model), then we accept it as a *correct* model for that particular APUF instance; otherwise, model $\mathbf{w}$ is a random model and it should be ignored. Note that $\mathbf{w}$ can match with at most one APUF instance, because if the model $\mathbf{w}$ corresponds to a particular APUF instance, then only this instance can have the matching probability ≥ 0.9 or ≤ 0.1, and for the other APUF instances, the matching probability would be around 0.5 due to the good uniqueness property of simulated APUF instances.

For the given $\mathcal{Q}$, we measure the noise rate of each APUF with respect to the challenges present in $\mathcal{Q}$. Subsequently, we rank the 10 APUF instances of 10-XOR APUF, using numbers between 1 to 10, with respect to their noise rate, i.e. the APUF with rank 1 has the highest noise rate. Let us denote the rank of the APUF, for which a correct model is built in the i-th run of CMA-ES, by k_i. Note that if a model $\mathbf{w}$ does not match with any of the APUFs in i-th run of CMA-ES, then we set $k_i = 0$. Thus, after 100 runs of CMA-ES, we have a set of rank values $\{k_1, \ldots, k_{100}\}$, and we count how many times each rank value appearing in this set $\{k_1, \ldots, k_{100}\}$. Table 1 shows the rank values and its corresponding counts.

From Table 1, it is evident that if an APUF instance is more noisy with respect to a given $\mathcal{Q}$, then the built model $\mathbf{w}$ can match with that APUF instance with high probability. In this case, rank 1 represents the APUF instances with highest noise value in each run of CMA-ES, and its count value is 36. If we estimate the probability of the 5 most noisy APUF instance, namely rank 1 to 5, the probability of having a correct is approximately 0.7 (70%).

The experimental result can be explained as follows. The challenge-reliability pairs $\mathcal{Q} = \{\mathbf{c}_i, R_i\}$ can be divided into two parts, i.e., reliable challenge-reliability pairs $\mathcal{Q}_r$ and noisy challenge-reliability pairs $\mathcal{Q}_n$ (see Section 2.3). For each APUF instance A_i, it has its own set of noisy challenge-reliability pairs $\mathcal{Q}_{i,n} \subset \mathcal{Q}_n, i = 0, \ldots, 9$. From the view of CMA-ES algorithm, it tries to converge to the APUF instance which has the largest number of pairs in the combined set $\mathcal{Q}_r \cup \mathcal{Q}_{i,n}$, i.e., highest Pearson correlation (see Step-4 of CMA-ES based attack on APUF in Section 2.3). Since $\mathcal{Q}_r$ is useful for modeling of all APUF instances, the set $\mathcal{Q}_{i,n}$ should be large to make the $\mathcal{Q}_r \cup \mathcal{Q}_{i,n}$ sufficient enough for modeling the i-th APUF instance. In other words, the CMA-ES tries to converge to APUF instance having largest value for $|\mathcal{Q}_{i,n}|/|\mathcal{Q}|$ (i.e. noise rate).

As a side remark, we built models of all the APUF instances in these 100 runs of CMA-ES.

3.2 Experiment-II

Objective of this experiment is to investigate how many different APUF models can be built if we execute CMA-ES many times using the *same* $\mathcal{Q}$.

In this case, we follow the same experiment setup as in Experiment-I with the exception that the same set $\mathcal{Q}$ is used for 100 runs of CMA-ES.

Result and Discussion. After the 100 runs of CMA-ES, we are able to model only five APUF instances out of 10 APUFs in 10-XOR APUF. This happened due to the probabilistic nature of CMA-ES, and the correlation between the built model and the noise rate which is shown in Experiment-I. Moreover, this experiment shows that the efficient approach for building all APUF models is to use different set $\mathcal{Q}$ in each run, i.e., the models of all PUFs are found in 100 runs when $\mathcal{Q}$ is randomly generated in each run (Experiment-I).

3.3 Experiment-III

Objective of this experiment is to investigate under which condition Becker's attack cannot build a model for a particular APUF in XOR APUF. We want to show that if the noise rate of APUF instances presenting at the XOR APUF's output are *not equally* (i.e., drawn from different distribution), then there are some APUF instances cannot be built by Becker's attack.

Setup. We perform the following extreme test. We execute the CMA-ES for APUF on 2-XOR APUF 100 times. The 2-XOR APUF consists of two APUF instances, namely A_0 and A_1, which are simulated using the Matlab as described in Section 5, and the noise rate of both APUFs are set at 1%. In each run, we use a newly generated set $\mathcal{Q}$ of size 70×10^3, and each challenge is evaluated 11 times. In this setup, we try to manipulate the reliability information presented in the final output, such that A_0 always has lower noise rate in the measured data set $\mathcal{Q}$. This is achieved by applying the majority voting of responses of A_0 before XOR-ing it with the response of A_1. In majority voting, we have tried with 5 and 10 measurements to observe the modeling performance. Let M be the number of measurements used in majority voting.

Result and Discussion. The experimental results are shown in Table 2. The second and the fourth columns refer to the counts of correct model building corresponding to A_0 and A_1, respectively, in 100 runs of CMA-ES. The third and the fifth columns describe the reliability of A_0 and A_1 measured just before XOR-ing, respectively, with mean (μ) and standard deviation (σ). From Table 2, it is prominent that if M is sufficiently large (≥ 10), then Becker's attack cannot build a model for A_0 as its reliability is enhanced by applying the majority voting compared to A_1.

Applying the majority voting for A_0 with a sufficiently large value for M, we can prevent the attacker to build a model for A_0 using Becker's attack, but this does not make the design secure, as the output of A_0 is exposed by XOR-ing the outputs of A_1 and 2-XOR APUF. Thus, eventually, the attacker can build a model for A_0 in straightforward way. But the lesson we learned from

Table 2: Results of Experiment-III

$M^{\dagger}$	A_0		A_1	
	Count	**Reliability** (μ, σ)	**Count**	**Reliability** (μ, σ)
5	8	(99.72%, 0.02)	92	(98.82%,0.05)
10	0	(99.58%, 0.02)	99	(98.78%,0.06)

† No. of measurements used in the majority voting of A_0.

this experiment is that if we can create two APUFs with two different noise rates, then the influence of reliability of A_0 and A_1 on the final output r of XOR APUF is also different or *unequal* in terms of noise. Even if the gap is very small as shown in Table 2, the attack fails to build a model of A_0. As studied in Experiment-I, if the noise of one APUF instance is always smaller than others, then that APUF is *hidden* from the attacker.

3.4 Lessons from the Experiments

After all experiments above, to successfully attack a PUF design based on APUFs the following conditions should be satisfied.

1. All APUF instances' outputs must have the same influence on the final output of PUF design.
2. The noise rate of all APUF instances should be similar.

In next section, we present the security and reliability analyses of the proposed MXPUF based on the insights of Becker's attack.

4 Security and Reliability Analyses of (x, y)-MXPUF

In this section, we analyze the security and reliability of our MXPUF design. In particular, we discuss how to select values for x and y, and where to insert the response of the upper layer x-XOR PUF in the intermediate challenge of the lower layer y-XOR PUF. We start with analyzing the role of challenge bit $\mathbf{c}[j]$ in a component APUF's output r, $j = 0, \ldots, n-1$. After this, we introduce and study a simplified version of a (x, y)-MXPUF, and then, extend this analysis to (x, y)-MXPUF.

4.1 Influence of challenge bit $\mathbf{c}[j]$ in APUF's output r

The influence of a challenge bit on APUF's response depends on its position in the challenge $\mathbf{c}$ [17,5,18]. From Eq. (1), it can be observed that $\mathbf{\Phi}[j+1], \ldots, \mathbf{\Phi}[n-1]$ do not depend on the challenge bit $\mathbf{c}[j]$. For a given challenge $\mathbf{c}$, based on the linear delay model of APUF, the delay difference Δ can be described as:

$$\Delta = (1 - 2\mathbf{c}[j]) \times \Delta_F + \Delta_N, \tag{8}$$

where $\varDelta_F = \sum_{i=0}^{j} \mathbf{w}[i]\frac{\mathbf{\Phi}[i]}{(1-2\mathbf{c}[j])}$, and $\varDelta_N = \sum_{i=j+1}^{n} \mathbf{w}[i]\mathbf{\Phi}[i]$.

One can notice that $\varDelta_N$ does not depend on $\mathbf{c}[j]$. Let $\varDelta_{j,0}$ be $\varDelta = \varDelta_F + \varDelta_N$ when $\mathbf{c}[j] = 0$ and r_0 be the corresponding response. Similarly, $\varDelta_{j,1}$ be $\varDelta = -\varDelta_F + \varDelta_N$ when $\mathbf{c}[j] = 1$, and r_1 be the corresponding response. If $\varDelta_{j,0} \times \varDelta_{j,1} \geq 0$ (or equivalently $|\varDelta_F| \leq |\varDelta_N|$), then it results in $r_0 = r_1$.

We follow existing PUF literature and assume that when generating an instance of an APUF all $\mathbf{w}_i$ are sampled from the same normal distribution $\mathcal{N}(0, \sigma^2)$ [13,12], and hence, $\varDelta_F \sim \mathcal{N}(0, (j+1) \times \sigma^2)$ and $\varDelta_N \sim \mathcal{N}(0, (n-j) \times \sigma^2)$. This implies that the probability $\Pr(r_0 = r_1)$ decreases with the increasing value of j, i.e.,

$$\Pr(r_0 = r_1) \approx (n-j)/n, j = 0, \ldots, n-1 \tag{9}$$

Thus, contribution of each challenge bit in security are not equal, which is an undesirable property.

This analysis is similar to that of Strict Avalanche Criterion (SAC) property of APUF discussed in [17,5,18]. The SAC property of a PUF refers to the relationship between the responses r and r' of two challenges $\mathbf{c}$ and $\mathbf{c}^i$, respectively, where $\mathbf{c}$ and $\mathbf{c}^i$ differ in the i-th bit. A PUF design with n-bit challenge is said to satisfy SAC property if $\Pr(r \neq r^i) = 0.5$ for all $i = 0, \ldots, n-1$. It is worth mentioning that the SAC property of APUF is not affected significantly by the reliability of APUF [18].

Now, we analyze the security and reliability of proposed MXPUF by exploiting the SAC property of APUF.

4.2 CASE-I: (1,1)-MXPUF

In $(1,1)$-MXPUF, the upper layer consists of a single component APUF A_u and the lower layer consists of a single component APUF A_l. Let us denote the responses to A_u and A_l by a and r, respectively. The final response of the $(1,1)$-MXPUF is response r of the lower layer.

Reliability of (1,1)-MXPUF In order to determine the effect of measurement noise, we evaluate a challenge $\mathbf{c}$ twice. Let (a_0, r_0) and (a_1, r_1) denote the corresponding outputs of A_u and A_l. We assume that the two APUFs A_u and A_l have the same noise rate β, i.e., $\Pr(a_0 \neq a_1) = \Pr(r_0 \neq r_1 | a_0 = a_1) = \beta$ where the probability is over the noise distribution—we actually assume the stronger statement that for any two manufactured APUFs A_u and A_l we have this property.

Suppose that $i+1$ is the feedback position of output a of A_u in the $(n+1)$-bit challenge of A_l. We derive

$$\begin{aligned}\Pr(r_0 \neq r_1) &= \Pr(r_0 \neq r_1 | a_0 = a_1)\Pr(a_0 = a_1) + \Pr(r_0 \neq r_1 | a_0 \neq a_1)\Pr(a_0 \neq a_1) \\ &= \beta(1-\beta) + \left(\frac{i+1}{n+1}\right)\beta. \end{aligned} \tag{10}$$

In practice, $\beta \ll 1$, and thus, (10) is approximately equal to $\beta + \beta\frac{i}{n}$.

We notice that our analysis shows that the reliability information of A_u and A_l available at the output of $(1,1)$-MXPUF are not *equal*: A_l contributes approximately β while A_u contributes approximately $\beta\frac{i}{n}$.

Security of $(1,1)$-MXPUF In this section, we discuss the security of the (1,1)-MXPUF with respect to Becker's attack and traditional modeling attacks. We also discuss a new type of attack on this structure to complete our analysis.

Becker's attack. Due to the unequal contribution of A_u and A_l to the reliability information extracted from the MXPUF, the model for A_u cannot be built as mentioned in Section 3.3. We performed an experiment where the position i at which a is inserted in the lower layer varies from 0 to $n-1$; our experiment confirms that the model of A_u cannot be built. To build the model of A_l, for a given challenge $\mathbf{c}$, we need to know the output a in order to compute the Δ for R' (see Eq. (6)). Since a cannot be accurately predicted by the attacker, modeling of A_l becomes impossible as well and Becker's attack fails for $(1,1)$-MXPUF.

Traditional modeling attack. $(1,1)$-MXPUF is not secure: For example, we can perform CMA-ES based modeling to learn A_u and A_l at the same time based on the CRPs of the $(1,1)$-MXPUF (instead of challenge reliability information pairs).

Linear approximation based modeling attack. We consider the following cryptanalysis attack on ($x \geq$ 1,1)-MXPUFs. We develop the attack based on the following observations:

1. The lower level y-XOR APUF consists of only one ($y=1$) APUF instance A_l, which has one more bit in its challenge which comes from the upper level x-XOR APUF.
2. The impact of a on the output of MXPUF depends on the feedback position of a as discussed in Section 4.1.

We *linearly approximate* the $(x,1)$-MXPUF, which we will denote by P, by using the $(n+1)$-bit APUF A_l with a substituted by 0. For a given challenge $\mathbf{c} = (\mathbf{c}[0], \ldots, \mathbf{c}[i], \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$, we compute:

$$p_{approx} = \Pr(A_l(\mathbf{c}') = P(\mathbf{c})),$$

where $\mathbf{c}' = (\mathbf{c}[0], \ldots, \mathbf{c}[i], 0, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$.

If p_{approx} is high, then the MXPUF **is accurately approximated** by APUF A_l which is vulnerable to the described modeling attacks: We can use a machine learning technique to build the model of A_l based on either CRPs or reliability information.

For the sake of analysis, we assume that all APUF components in the $(x,1)$-MXPUF are a 100% reliable, and the output a of the x-XOR APUF is uniform, i.e., $\Pr(a=0) = \Pr(a=1) = 1/2$. Then,

$$\begin{aligned} p_{approx} &= \Pr(A_l(\mathbf{c}') = P(\mathbf{c})) \\ &= \Pr\left(A_l(\mathbf{c}') = P(\mathbf{c})|a=0\right)\Pr(a=0) + \Pr(A_l(\mathbf{c}') = P(\mathbf{c})|a=1)\Pr(a=1) \\ &= 1 \times 1/2 + \frac{n-i}{n} \times 1/2 = 1/2 + \frac{n-i}{2n}. \end{aligned} \tag{11}$$

Equation (11) shows hat p_{approx} decreases while i increases. We performed an experiment with $i = 0, n/2$ and $n-1$ for the $(1,1)$-MXPUF, which we approximated by a single APUF as described above. The experiment tells us that p_{approx} can be around $97\%, 77\%$ and 50% when $i = 0, n/2$ and $n-1$, respectively. This experimental result matches Eq. (11). We notice that the same argument can be used to show that a (x,y)-MXPUF can be approximated by a single y-XOR APUF.

4.3 CASE-II: (x,y)-MXPUF

Based on the above analysis of $(x \geq 1, 1)$-MXPUF, we can now explain how to choose the parameters x, y and feedback position i of a.

Security Analysis We will show: i) MXPUF is secure against Becker's attack and traditional modeling attacks, ii) the importance of the parity of y, and iii) why a should be inserted at the middle of challenge $\mathbf{c}$.

Becker's attack The security argument for the $(1,1)$-MXPUF copies over to the (x,y)-MXPUF in that Becker's attack fails, i.e., the adversary cannot build models for the x component APUFs in the x-XOR APUF if a is inserted at the middle position. She cannot build models of the y component APUFs in the y-XOR APUF as a cannot be predicted. As shown in [6], the adversary can change the environmental condition to get more 'noisy' challenges and thus, the efficiency of the attacks in [2] can be improved in terms of the number of CRPs used. However, the combined attacks of [2] and [6] will still not work because of the same argument.

Linear approximation based modeling attack As discussed above, we can approximate (x,y)-MXPUF by a y-XOR APUF. It is evident that when y increases, then the prediction accuracy of the approximated y-XOR PUF decreases. For $y = 2$ and the feedback position at the middle (i.e., $p_{approx} = 75\%$), the prediction accuracy of approximated model of $y(=2)$-XOR PUF is equal to $p^2_{approx} = 75\% \times 75\% = 56\%$, and it is the upper bound for this attack when $y \geq 2$ since introducing more APUF instances in y-XOR APUF decreases the prediction accuracy of final model. This implies that for $y \geq 2$, (x,y)-MXPUF is secure against this attack.

Traditional modeling attacks When the two aforementioned modeling attacks are ruled out, the only other known way to attack is to build models of all $(x+y)$ component APUFs at the same time using traditional methods. Let $b_0, b_1, \ldots, b_{y-1}$ be the outputs of the y APUFs in the lower layer y-XOR PUF for a substituted by 0. Since traditional modeling attacks work best if no measurement noise is present, we assume no measurement noise at all. This means that Section 4.1 is applicable: For a given challenge $\mathbf{c}$, b_i depends on the upper layer output a (in that bit b_i would flip if a would be substituted by 1) with probability $p = (i+1)/(n+1) \approx i/n$ if the feedback position of a is i (the probability is taken over the possible manufacturing variations which produce each of the component APUFs). For a given challenge $\mathbf{c}$, let p_r be the probability that the response of (x, y)-MXPUF is equal to $r = 1 \oplus b_0 \oplus \ldots \oplus b_{y-1}$ if a would be substituted by 1. I.e., p_r is the probability that a affects the the final response r. Then p_r depends on i:

$$p_r = \sum_{k=1, k \text{ odd}}^{y} \binom{y}{k} p^k (1-p)^{y-k} = \frac{1-(1-2p)^y}{2}. \tag{12}$$

From Eq. (12) we infer that, for odd y, the influence of a on r reduces when the feedback position of a is towards the least significant challenge bit position. For even value of y, the influence of a increases when feedback position of a drifts from the very last bit of challenge to the middle bit position and then it decreases when moving a from the middle position to the very first bit challenge.

From the next discussion on reliability and experiments we obtain that a should be inserted at the middle position and $y = 3$ (it must be ≥ 2 in order not to be vulnerable to the linear approximation based modeling attack). This gives a $(x, 3)$-MXPUF with the feedback position at the middle. Compared to an $(x+3)$-XOR APUF, it has almost similar security against modeling attacks such as LR and CMA-ES [20,27]. Moreover, it is secure against Becker's attack.

Reliability analysis From the above security analysis, we know that y must be ≥ 2 in order not to be vulnerable to the linear approximation based modeling attack. For best reliability (least influence of measurement noise), $y = 3$ is the optimal choice.

We now compare the reliability of $(x, 3)$-MXPUF and $(x+3)$-XOR APUF. Let β_x and β_y be the noise rates of the x-XOR APUF and $(y=3)$-XOR APUF. Then the noise rate of the $(x+3)$-XOR APUF is $\beta_x(1-\beta_y) + \beta_y(1-\beta_x) = \beta_x + \beta_y - 2\beta_x\beta_y$. If $\beta_y \ll \beta_x < 1$ when $y = 3 \ll x$, the noise is approximated by $\beta_x + \beta_y$. We can further approximate it by using the fact $\beta_y \ll \beta_x$, i.e., the noise rate of $(x+2)$-XOR APUF is equal to β_x.

Actually, the noise of MXPUF can be computed as $\beta_y + \beta_x \times p_r \approx \beta_x \times \frac{1-(1-2p)^y}{2}$ where $p = i/n$ and i is the feedback position of a. The noise rate of $(x, 3)$-MXPUF can be approximated as $\beta_x/2$ when a is inserted at the middle position (i.e., $i = n/2$ and $p = 1/2$), and thus, it is equal to half of that of $(x+3)$-XOR APUF. We confirm these approximations of the noise rates through experiments in Section 5.1.

5 Simulation Results

We simulated the APUF variants using Matlab (for challenge sizes of 64-bit) assuming that propagation delay of each delay component follows normal distribution with $\mu = 10$ and $\sigma = 0.05$. In case of simulation, we directly compare the delays of top and bottom paths using an ideal comparator, so there is no impact of the non-ideal arbiter component on the PUF behavior.

To evaluate the reliability metric of simulated APUF (as well as MXPUF and XOR APUF), we have considered a simulated additive noise (in case of hardware implementation this noise occurs due to temperature and supply voltage variations) with the delay distribution of delay components. In the presence of noise, each delay component follows a normal distribution $\mathcal{N}(\mu + 0, \sigma^2 + \sigma^2_{\text{noise}})$, where noise distribution is a normal distribution $\mathcal{N}(0, \sigma^2_{\text{noise}})$. In our simulations, we have used following relation between σ and σ_{noise} to control the reliability levels: $\sigma_{\text{noise}} = \gamma\sigma$, where $0 \leq \gamma \leq 1$. For $\gamma = 0$, PUF instance is 100% reliable. In case of hardware implementation on FPGA and ASIC, it not necessary to know the value of γ as one can control the reliability level by changing the operating temperature and supply voltage.

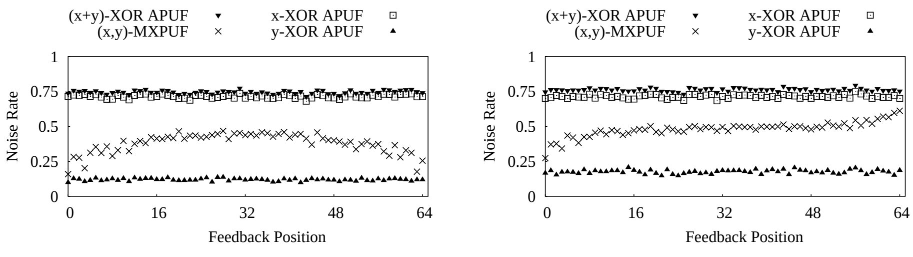
In this section, we do experiments to confirm our theoretical findings about security and reliability of (x, y)-MXPUF

5.1 Reliability

To compare the reliability, we have simulated $(x+y)$-XOR APUF, (x, y)-MXPUF, x-XOR APUF and y-XOR APUF for $x = 20$, and $y = 2$ and $y = 3$. In simulation, we have used 64-bit APUFs, and the noise rate of each APUF is around 5%. To estimate the reliability, we evaluated each PUF design with 10×10^3 randomly generated challenges, and each challenge is measured 11 times to determine whether it is noisy or not. If the repeatability of a challenge is 100%, we say it is reliable; otherwise, it is a noisy challenge. The reliability of a PUF is estimated as the fraction of reliable challenges.

We varied the feedback position of a from 0 to 64, and for each feedback position, we measured the reliability of $(x + y)$-XOR APUF, (x, y)-MXPUF, x-XOR APUF and y-XOR APUF. The simulated results are presented in Fig. 3.

Figures 3a and 3b show that the noise rate of $(x + y)$-XOR APUF and x-XOR APUF are close to each other as the value of y is very small compared to x (i.e. $y = 2$ or $y = 3$). The noise rate of y-XOR APUF is very small compared to $(x + y)$-XOR APUF and x-XOR APUF. The noise rate of $(x, 3)$-MXPUF increases when the feedback position increases from 0 to 64. However, the noise rate of $(x, 2)$-MXPUF reaches the maximum value at feedback positions around 32. This is because of the parity of y (see Eq. (12)). The noise rate $(x, 3)$-MXPUF is equal to half of the noise rate of $(x+3)$-XOR APUF when a is fed at the middle. The experiments confirm our findings related to reliability (see Section 4).



(a) $x = 20$ and $y = 2$ (b) $x = 20$ and $y = 3$

Fig. 3: Noise rate of (x+y)-XOR APUF, (x,y)-MXPUF, x-XOR APUF and y-XOR APUF when noise rate of APUF is around 0.05.

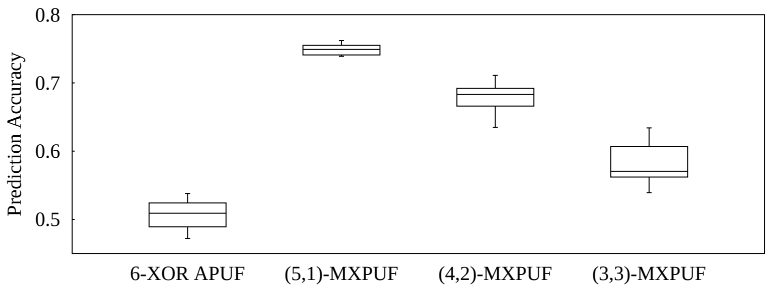


Fig. 4: Modeling of $(x + y)$-XOR APUF and (x, y)-MXPUF using CMA-ES and CRPs of PUFs. Each boxplot is built using 10 prediction accuracy values of 10 different models of a PUF design.

5.2 Security

In this section, we present the security comparison between simulated $(x + y)$-XOR APUF and (x, y)-MXPUF against modeling attack. Note that in this experiment, all APUF instances are reliable, i.e., we do not consider Becker's attack. Since the linear delay model of MXPUF is not known, we have used the CMA-ES with the CRPs for modeling MXPUF and XOR APUF. Our target is to compare the security between XOR APUF and MXPUF, we can use small APUFs, i.e., 20-bit APUFs. In modeling, we used the training and testing sets of 20×10^3 CRPs and 1000 CRPs, respectively. In each run of CMA-ES algorithm, we set the number of iteration to 1000. We built the models for 6-XOR APUF, $(5, 1)$-, $(4, 2)$- and $(3, 3)$-MXPUFs. We repeated the attack on each PUF 10 times, and in each attack, the new set of training CRPs is randomly generated. The prediction accuracy values of our attacks on these PUFs are reported in Fig. 4. This figure shows that the security of (3,3)-MXPUF is slightly weaker than that of 6-XOR APUF. The security of (4,2)-MXPUF is poor compared to (3,3)-MXPUF due to the value of y (see Eq. (12)). Since the output of MXPUF fully depends on the APUF instances in y-XOR PUF and partially depends on the APUF instances in x-XOR APUF, the security of (x, y)-MXPUF would increase with increasing value of y. In fact, the result of $(4, 2)$-MXPUF is acceptable in terms of modeling attack, since a successful modeling attack is expected to have prediction accuracy larger than 90% [28]. Moreover, the security of $(5, 1)$-MXPUF is poorer compared to others. This is because of linear approximation based attack mentioned in Section 4.

6 Conclusion

In this paper, we comprehensively studied the Becker's attack based on reliability information [2], and then, we proposed a new APUF composition (called MXPUF). The MXPUF is secure against both the Becker's attack [2] and traditional modeling attacks [20,27]. Compared to XOR APUF design, the security of MXPUF against traditional modeling attack is slightly weaker than XOR APUF while the same number of APUFs are used in the compositions. Moreover, the reliability of (x, y)-MXPUF is twice the reliability of $(x + y)$-XOR APUF. We verified all our findings through simulated PUFs.

Appendix

A Enhanced Repeatability based CMA-ES on APUF

In the reliability based CMA-ES modeling, the efficiency of the model depends on how the reliability (R) and hypothetical reliability (R') values are defined. Let us denote the reliability R in Eq. (5) by H_1 and hypothetical reliability R' in Eq. (6) as H'_1. The fitness value of a model is defined as the correlation between H_1 and H'_1. In this section, we propose two alternative definitions for (H_1, H'_1), namely, (H_2, H'_2) and (H_3, H'_3), and subsequently, we show that they can build an APUF model using lesser number of CRPs than that required for modeling using (H_1, H'_1). The (H_2, H'_2) and (H_3, H'_3) are defined as follows:

$$H_2 = |M/2 - \sum_{i=1}^{M} r_i| \qquad \text{and} \qquad H'_2 = |\Delta| \tag{13}$$

$$H_3 = \frac{1}{M}\sum_{i=1}^{M} r_i \qquad \text{and} \qquad H'_3 = \Phi(-\Delta/\sigma_N) \tag{14}$$

Like the parameter ϵ in the modeling of APUF using (H_1, H'_1), the parameter σ_N is to be learned while (H_3, H'_3) is used. Thus, in both the cases (H_1, H'_1) and (H_3, H'_3), total number of parameters to be learned for an n-bit APUF is $(n+2)$. However, in the case of (H_2, H'_2) only $(n + 1)$ parameters $\mathbf{w}[0], \ldots, \mathbf{w}[n]$ is to be learned.

The only difference between (H_1, H'_1) and (H_2, H'_2) is $H'_1 \neq H'_2$. The disadvantage of H'_1 is that it exploits transformed digital value of Δ based on the

Table 3: A comparison of 64-bit APUF's modeling accuracy using CMA-ES

$N^\dagger$	**Modeling Accuracy(%)**		
	(H_1, H_1')	(H_2, H_2')	(H_3, H_3')
600	60.33	78.27	96.02
1500	69.60	96.64	97.80
3000	97.49	97.65	98.38
6000	97.85	97.98	98.40

† No. of CRPs is used to train a model.

parameter ϵ, while the values of H_1 lies in $[0, M/2]$. This results in less correlation between H_1 and H_1'. On the other hand, in H_2', we have directly used absolute value of $\varDelta$ (i.e. $|\varDelta|$), and in this case, H_2 and H_2' exhibit better correlation compared to (H_1, H_1').

In H_1 and H_2, the polarity of response (i.e., whether it is 0 or 1) is not considered while computing reliability information R. However, H_3 retains the response polarity information within the reliability information as reliability information lies in $[0, 1]$. In this case, we have borrowed the reliability formulation shown in Eq. (3) to define H_3'. Since both the response polarity and reliability information are used in mode building, (H_3, H_3') yields a high accuracy APUF model using less number of CRPs.

Table 3 depicts the modeling accuracy of 64-bit simulated APUF using CMA-ES and reliability information. In this simulation, we have followed the same setup as mentioned Section 5 with $\sigma_{\text{noise}} = \sigma/10$. For this value of σ_{noise}, the reliability of APUF instances are around $96 - 97\%$. From Table 3, it is evident that CMA-ES modeling using our proposed (H_2, H_2') and (H_3, H_3') outperforms the modeling using (H_1, H_1') as proposed in [2]. In addition, (H_3, H_3') outperforms (H_2, H_2') as both the response polarity and reliability information are considered in (H_3, H_3'). Although (H_3, H_3') is better than (H_2, H_2') for modeling a standalone APUF, in the context of XOR APUF, we have observed from experimental result that the performance of (H_1, H_1') and (H_2, H_2') are superior than (H_3, H_3'). One reason is the consideration response polarity in the H_3 as it makes the modeling task more difficult in the context of XOR APUF. Next, we discuss the performance of (H_1, H_1') and (H_2, H_2') in the context of XOR APUF modeling.

B Enhanced Repeatability based CMA-ES on XOR APUF

In Section 3, we have discussed the XOR APUF modeling using CMA-ES and (H_1, H_1'). The (H_1, H_1') was proposed by Becker in [2], and it works well for XOR APUF. However, our proposed (H_2, H_2') can achieved better modeling accuracy for XOR APUF using less number of $(\mathbf{c}, R)$ pairs compared to (H_1, H_1').

Table 4: Modeling results of 4-XOR APUF using CMA-ES and different reliability models

Model setup	$N^\dagger$	**Modeling Acc.(%)**				**Frequency***			
		A_0	A_1	A_2	A_3	A_0	A_1	A_2	A_3
(H_1, H'_1)	10×10^3	65.10	66.17	67.40	68.96	0	0	0	0
	20×10^3	98.08	97.91	98.23	98.33	1	4	3	1
	30×10^3	98.27	98.06	98.27	98.39	19	10	6	3
	50×10^3	98.31	98.16	98.37	98.44	39	20	17	10
(H_2, H'_2)	10×10^3	97.53	97.09	97.10	95.24	22	24	31	8
	20×10^3	97.86	97.75	97.74	97.78	24	29	29	17
	30×10^3	98.08	97.89	98.02	98.12	47	27	20	6
	50×10^3	98.17	98.06	98.23	98.27	50	29	16	5

† No. of CRPs is used to train an APUF as well as 4-XOR APUF models.
* No. of correct models (prediction accuracy $> 90\%$) for A_i out of 100 runs of CMA-ES.

To demonstrate the performance of (H_2, H'_2), we have simulated a 4-XOR APUF in Matlab by following the same simulation setup as mentioned Section 5 with $\sigma_{\text{noise}} = \sigma/10$. For this value of σ_{noise}, the reliability of APUF instances are around $96 - 97\%$. Table 4 shows the performance comparison of (H_1, H'_1) and (H_2, H'_2) based on a simulated 4-XOR APUF modeling. In Table 4, we have reported two aspect: i) modeling accuracy, and ii) frequency of the correct APUF models (a correct model has prediction accuracy greater than 90). We have run CMA-ES for 100 times, and it can be observed that (H_2, H'_2) results more successful models than that of (H_1, H'_1). Let us consider the case of $N = 10 \times 10^3$. In this case, there is no successful model for APUFs if (H_1, H'_1) is used. However, the attacker can build models for APUF using (H_2, H'_2) with $N = 10 \times 10^3$. Although (H_2, H'_2) is derived from (H_1, H'_1) with a small change in H'_1, this modification results in an efficient modeling of XOR APUF.

References

1. Becker, G.T.: On the Pitfalls of using Arbiter-PUFs as Building Blocks. IACR Cryptology ePrint Archive 2014, 532 (2014)
2. Becker, G.T.: The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In: Proc. of 17th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) (2015)
3. Brzuska, C., Fischlin, M., Schrauder, H., Katzenbeisser, S.: Physically Uncloneable Functions in the Universal Composition Framework. In: Rogaway, P. (ed.) CRYPTO, pp. 51–70 (2011)
4. Delvaux, J., Verbauwhede, I.: Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise. In: IEEE 6th Int. Symposium on Hardware-Oriented Security and Trust (2013)

5. Delvaux, J., Gu, D., Schellekens, D., Verbauwhede, I.: Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible? In: Proc. of 16th International Workshop on Cryptographic Hardware and Embedded Systems (CHES). pp. 451–475 (2014)
6. Delvaux, J., Verbauwhede, I.: Fault Injection Modeling Attacks on 65 nm Arbiter and RO Sum PUFs via Environmental Changes. IEEE Trans. on Circuits and Systems 61-I(6), 1701–1713 (2014)
7. Ganji, F., Tajik, S., Fäßler, F., Seifert, J.P.: Strong Machine Learning Attack against PUFs with No Mathematical Model. In: CHES. pp. 391–411. Springer Berlin Heidelberg (2016)
8. Ganji, F., Tajik, S., Seifert, J.P.: Why attackers win: on the learnability of XOR arbiter PUFs. In: International Conference on Trust and Trustworthy Computing. pp. 22–39. Springer International Publishing (2015)
9. Ganji, F., Tajik, S., Seifert, J.P.: PAC learning of arbiter PUFs. Journal of Cryptographic Engineering 6(3), 249–258 (2016)
10. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM CCS (2002)
11. Kumar, S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: Extended abstract: The butterfly PUF protecting IP on every FPGA. In: HOST. pp. 67–70 (June 2008)
12. Lao, Y., Parhi, K.K.: Statistical Analysis of MUX-Based Physical Unclonable Functions. IEEE Trans. on CAD of Integrated Circuits and Systems 33(5), 649–662 (2014)
13. Lim, D.: Extracting Secret Keys from Integrated Circuits. Master's thesis, MIT, USA (2004)
14. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13(10), 1200–1205 (October 2005)
15. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In: Sadeghi, A.R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, pp. 3–37. Information Security and Cryptography, Springer, Berlin Heidelberg (2010)
16. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing Techniques for Hardware Security. In: Proc. of IEEE International Test Conference(ITC). pp. 1–10 (Oct 2008)
17. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for Design and Implementation of Secure Reconfigurable PUFs. ACM Trans. Reconfigurable Technol. Syst. 2(1), 1–33 (2009)
18. Nguyen, P.H., Sahoo, D.P., Chakraborty, R.S., Mukhopadhyay, D.: Security Analysis of Arbiter PUF and Its Lightweight Compositions Under Predictability Test. ACM TODAES 22(2), 20 (2016)
19. Pappu, R.S.: Physical one-way functions. Ph.D. thesis, Massachusetts Institute of Technology (March 2001)
20. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Proc. of 17th ACM conference on Computer and communications security(CCS). pp. 237–249. ACM, New York, NY, USA (2010)
21. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Majzoobi, M., Koushanfar, F., Burleson, W.P.: Efficient Power and Timing Side Channels for Physical Unclonable Functions. In: Proc. of 16th International Workshop on Cryptographic Hardware and Embedded Systems (CHES). pp. 476–492 (2014)

22. Sahoo, D.P., Saha, S., Mukhopadhyay, D., Chakraborty, R.S., Kapoor, H.: Composite PUF: A New Design Paradigm for Physically Unclonable Functions on FPGA. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). Arlington, VA, USA (May 2014)
23. Sölter, J.: Cryptanalysis of Electrical PUFs via Machine Learning Algorithms. Master's thesis, Technische Universität München (2009)
24. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC. pp. 9–14 (2007)
25. Tajik, S., Lohrke, H., Ganji, F., Seifert, J.P., Boit, C.: Laser Fault Attack on Physically Unclonable Functions. In: 12th Workshop on Fault Diagnosis and Tolerance in Cryptography (FTDC) (2015)
26. Tajik, S., Dietz, E., Frohmann, S., Seifert, J., Nedospasov, D., Helfmeier, C., Boit, C., Dittrich, H.: Physical Characterization of Arbiter PUFs. In: Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings. pp. 493–509 (2014)
27. Tobisch, J., Becker, G.T.: On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation. In: Proc. of 11th International Workshop on Radio Frequency Identification: Security and Privacy Issues (RFIDsec). pp. 17–31 (2015), http://dx.doi.org/10.1007/978-3-319-24837-0_2
28. Vijayakumar, A., Kundu, S.: A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics. In: DATE. pp. 653–658 (2015)
29. Yu, M.D.M., M'Raïhi, D., Sowell, R., Devadas, S.: Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In: CHES. pp. 358–373 (2011)