# The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks

Phuong Ha Nguyen[†], Durga Prasad Sahoo[‡], Chenglu Jin[†], Kaleel Mahmood[†], Ulrich Rührmair[§], and Marten van Dijk[†⋆]

[†] University of Connecticut, USA
[‡] Bosch India (RBEI/ESY), India
[§] Horst Görtz Institute for IT-Security, Ruhr Universität Bochum, Germany

**Abstract.** Silicon Physically Unclonable Functions (PUFs) have been proposed as an emerging hardware security primitive in various applications such as device identification, authentication and cryptographic key generation. Despite their potential, PUF designs based on the Arbiter PUF (APUF) are vulnerable to classical machine learning attacks, which use challenge response pairs. Classical machine learning can be mitigated in the $x$-XOR APUF when enough APUF components have been employed (high $x$). However, reliability based machine learning attacks cannot be prevented by increasing $x$. In this paper, we study the most prominent reliability based machine learning attack, the CMA-ES reliability attack. This work is the first to provide analysis and experimentation to explain under which conditions the CMA-ES reliability attack succeeds and where it fails. Based on these insights, we develop two key contributions. First, we demonstrate how the accuracy of the CMA-ES reliability attack can be improved through enhanced modeling. Second, we propose a new PUF design, the $(x, y)$-Interpose PUF. Through theory and simulation, we show our new PUF model is not vulnerable to the CMA-ES reliability attack, classical machine learning attacks and special attacks that approximate the Interpose PUF as an XOR APUF. In addition, we determine that the security of the IPUF can be reduced to the security of an XOR APUF under classical machine learning attacks, whose complexity depends exponentially on the number of component APUFs in the XOR APUF as shown in the literature. We also show our proposed $(x, y)$-Interpose PUF design is twice as reliable

[⋆] The IPUF design (formerly called MXPUF) was first proposed in the e-print, "MXPUF: Secure PUF Design against State-of-the-art Modeling Attacks", by a subset of current author list. However, this paper has significantly more content and new material as compared to the original paper. This paper presents new simulation results for machine learning attacks on a larger range of IPUF configurations and simulation results for the strict avalanche criterion (SAC) property. This paper also presents experimental results for an IPUF FPGA implementation as well as an analysis of the security issues related to APUF FPGA implementation. Lastly, almost all of the original paper has been rewritten with more text and additional figures to clearly explain the theory behind the IPUF design. The source code for this paper is publicly available at Github: Defense Attack (DA) PUF Library.

as an $(x + y)$ XOR APUF while using the same hardware overhead as an $(x + y)$ XOR APUF.

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

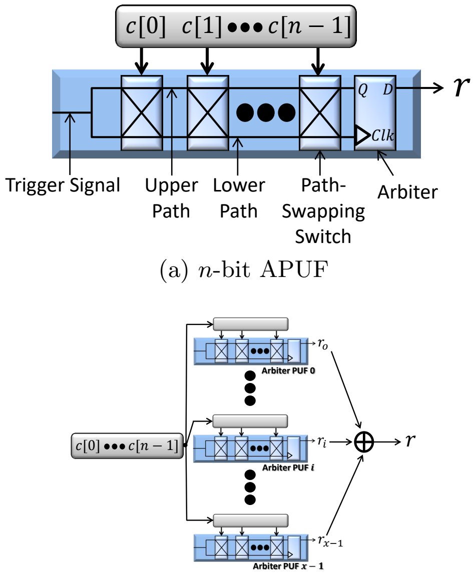
## 1 Introduction

Intuitively, a silicon Physically Unclonable Function (PUF) [12] is a fingerprint of a chip which behaves as a one-way function in the following way: it leverages process manufacturing variation to generate a unique function taking "challenges" as input and generating "responses" as output. The function is *Hardware Unclonable* in that it cannot be cloned in hardware. This means the PUF's internal behavior, e.g. its unique physical characteristics or behavior of its wires, cannot be read out accurately enough. Hardware unclonability also means it is not feasible to manufacture two PUFs with the same responses to a significant subset of challenges. A PUF is *Software 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 [25]. Silicon PUFs have been proposed for use in a wide variety of applications including device identification and authentication [20], binding software to hardware platforms [15], secure storage of cryptographic secrets [35], and secure protocol design [4].

Fig. 1a depicts an Arbiter PUF (APUF) [12,17,18] with challenge input vector $\mathbf{c} \in \{0,1\}^n$ and response bit $r \in \{0,1\}$. An APUF is considered a strong PUF due to the large $(2^n)$ number of challenge response pairs (CRPs) which is infeasible to enumerate. Fig. 1b depicts an XOR APUF [30] which XORs the response bits of different APUFs to produce a final output response bit. APUF variants and Lightweight Secure PUFs [22] 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 mathematical machine learning based modeling attacks [26,28,33,10,11,9,1].

A hierarchy of the different types of machine learning attacks on APUFs, XOR APUFs and IPUFs is shown in Fig. 2 and the vulnerabilities of different PUF designs to these attacks are shown in Fig. 3. In this paper, the types of machine learning attacks that use CRPs as training data are referred to as classic machine learning attacks. Classical machine learning attacks can be further sub-divided into two techniques: black box methods which operate without an exact mathematical PUF model and white box methods which learn a PUF's parameters when given an exact mathematical model. Since white box methods exploit more information about the PUF (i.e., mathematical PUF model), white box methods are much more efficient than black box methods .



(a) $n$-bit APUF

(b) $x$-XOR APUF

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

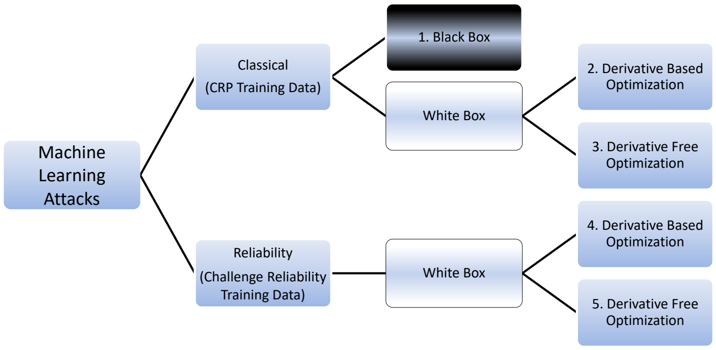
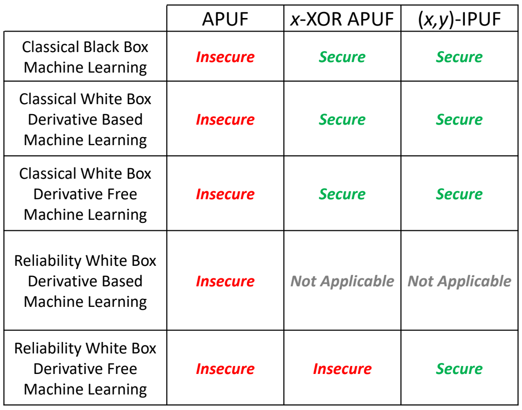

Machine Learning Attacks
Classical
(CRP Training Data)
1. Black Box
White Box
2. Derivative Based Optimization
3. Derivative Free Optimization
Reliability
(Challenge Reliability Training Data)
White Box
4. Derivative Based Optimization
5. Derivative Free Optimization

Fig. 2: Hierarchy of machine learning attacks on arbiter based PUFs, i.e., APUF, $x$-XOR PUF and $(x, y)$-IPUF.

White box classical machine learning was the first type of machine learning attack to model PUFs[12]. One solution to counter PUF's vulnerabilities to classical machine learning attacks (both white and black box types) is to increase the complexity of the PUF. In the $x$-XOR APUF this can be done by increasing the number of APUF components $x$. Linearly increasing $x$ causes the classical machine learning attack to require exponentially more training data and time complexity [26], making them infeasible.



| | APUF | *x*-XOR APUF | (*x*,*y*)-IPUF |
|---|---|---|---|
| Classical Black Box Machine Learning | *Insecure* | *Secure* | *Secure* |
| Classical White Box Derivative Based Machine Learning | *Insecure* | *Secure* | *Secure* |
| Classical White Box Derivative Free Machine Learning | *Insecure* | *Secure* | *Secure* |
| Reliability White Box Derivative Based Machine Learning | *Insecure* | *Not Applicable* | *Not Applicable* |
| Reliability White Box Derivative Free Machine Learning | *Insecure* | *Insecure* | *Secure* |

Fig. 3: Known machine learning attacks on APUFs, $x$-XOR PUFs and $(x, y)$-IPUFs. Arbiter PUFs are vulnerable to all the attack types [26,5,3]. When $x$ is large enough, $x$-XOR Arbiter PUFs are secure against all classical attacks [26,33] but insecure against white box derivative free reliability based machine learning attacks [3]. IPUFs are secure against all known machine learning attacks.

On the bottom branch of Fig. 2 we have machine learning attacks which use reliability information for training data, referred to as reliability based machine learning attacks. Currently there are no known black box reliability based machine learning attack. White box reliability based machine learning is further broken down into techniques that use derivatives like Least Squares [5] and derivative free approaches like CMA-ES. Thus far, reliability based machine learning with derivatives has only been applied to Arbiter PUFs.

The first efficient derivatie free white box reliability based machine learning attack on an $x$-XOR APUF and an $x$-way XOR APUF was demonstrated by Becker [3]. This attack uses CMA-ES to optimize its mathematical model, hence we refer to it in this paper as the CMA-ES reliability attack. In the CMA-ES reliability attack, each challenge is measured multiple times to generate reliability information. Instead of using CRPs directly for building the component APUF models, challenge reliability pairs are used. Due to the nature of the reliability measurements, in this attack each APUF in the $x$-XOR APUF can be modeled individually. Unlike classical machine learning attacks, for the CMA-ES reliability attack increasing $x$ in an $x$-XOR APUF only linearly increases the amount of training data required to model the PUF. In short, the CMA-ES reliability attack makes the $x$-XOR APUF highly vulnerable to modeling, even when $x$ is large [3].

One trivial approach to prevent white box derivative free reliability based machine learning attacks is to make the reliability of the PUF 100% as described in the LPN-PUF design [14]. However, the LPN-PUF requires a large hardware overhead which does not make the LPN-PUF a light-weight solution. In [34], the authors suggest enhancing the reliability of an $x$-XOR APUF using majority voting to make reliability based machine learning attacks more difficult. By definition this approach does not actually stop reliability based machine learning attacks. Moreover, this design is not efficient in terms of throughput and hardware overhead. A more detailed comparison to the majority voting XOR APUF is given in Section 8.

Overall the IPUF addresses the core of silicon PUF design, how one can construct a silicon PUF whose responses are generated by analog computing on manufacturing variations with post processing (inside the trusted computing base) in the form of only a couple of digital gate computations after analog to digital conversion. Regarding PUFs, in this paper we offer three significant contributions.

1. We improve the original CMA-ES reliability attack by using more precise mathematical comparisons between the reliability of the estimated model and the measured reliability of the PUF we are attacking. By using this new technique we are able to model both the APUF and $x$-XOR APUF more accurately than the original CMA-ES reliability attack. We then experiment and analyze the conditions under which the CMA-ES reliability attack operates.
2. Based on these findings we propose a new PUF design called the Interpose PUF (IPUF) (see Figure 4). We show that the IPUF is secure against reliability based machine learning attacks. Due to this security, the only way to attack an IPUF is through approximating it as an XOR APUF or through classical machine learning attacks.
   Regarding classical machine learning attacks, we prove that derivative based white box machine learning reduces to attacks that approximate the IPUF as an XOR APUF. We further show that these types of approximation attacks can be nullified through careful choice of the IPUF design parameters. Therefore, only classical white box machine learning that is derivative free is applicable. This makes the IPUF stronger than an XOR APUF design because it requires fewer APUF components to make it secure against classical machine learning (since more efficient derivative based techniques are not usable).
3. We provide open source code for all of our simulations and experiments. This includes code for machine learning attacks on APUF, XOR APUF and IPUF in Matlab and C#. Code to create various PUF models on FPGA hardware is also given.

Our paper is organized as follows: In Section 2 we cover the basic delay based and reliability based APUF models, as well as the design of the IPUF. In Section 3 we discuss the main machine learning attacks on PUFs. In Section 3 we also develop an approximation attack specifically for use against the IPUF and we explain the new modeling approach to enhance the original CMA-ES reliability attack. We further analyze the conditions under which the CMA-ES reliability attack functions in Section 4. Based on these findings we explain how the IPUF can mitigate the CMA-ES reliability attack in Section 5. In Section 5 we also analyze other pertinent properties of the IPUF. These properties include the security of the IPUF against the CMA-ES reliability attack, classical machine learning attacks and our proposed approximation attack, as well as the reliability of the IPUF. We provide comprehensive experimental results to reinforce our claims about the IPUF's security and reliability in Section 6. In this section we also experimentally demonstrated the improved accuracy of our enhanced CMA-ES reliability attack on APUF and XOR APUFs. Lastly, we offer concluding remarks in Section 9.

## 2 The Arbiter PUF and IPUF

In this section, we briefly introduce the analytical delay model [17] and reliability model [5] of the APUF. We also discuss the basic design of the newly proposed $(x, y)$-IPUF. Descriptions of both designs are necessary to understand the effectiveness of machine learning attacks on PUFs in Section 3.

### 2.1 The Arbiter PUF Linear Additive Delay Model

In [17,18], an analytical model called the *Linear Additive Delay Model* was presented. As shown in [17], 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: $r = 0$ if $\Delta \geq 0$. Otherwise, $r = 1$.

In [17,29,26] it was shown that APUFs are vulnerable to classical machine learning attacks. As a result, the $x$-XOR Arbiter PUF (XOR APUF) was introduced in [30] (See Fig 1b). Due to the non-linearity introduced by XOR operation, classical machine learning attacks can become impractical if a sufficient number of APUFs are employed in the $x$-XOR APUF [26,33].

### 2.2 The Arbiter PUF Reliability Model

Typically, manufacturing variability and measurement noise are two undesired phenomenons in an electric circuit [5]. Variability is caused by the manufacturing processes and noise is a random temporal phenomenon. Examples of noise include instability of the supply voltage or a change of circuit temperature. While noise is a non-reproducible physical feature, variability is reproducible and is exploited to build PUFs.

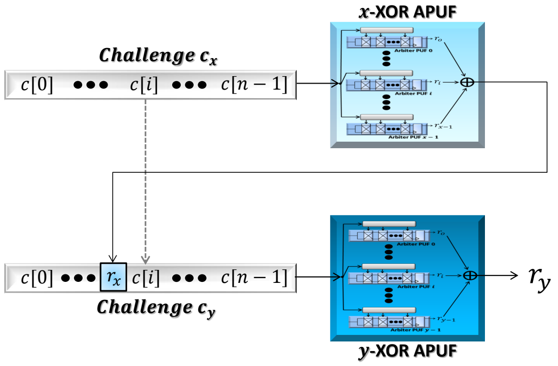


Fig. 4: The $(x, y)$-IPUF (Interpose PUF) is comprised of an $x$-XOR APUF whose input $r_x$ is interposed in the input of a $y$-XOR APUF.

Due to noise, the reproducibility or reliability of the output of the PUF is not perfect, i.e., applying the same challenge to a PUF will not produce a response bit with the same value every time. 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 [5].

In an APUF we can measure the (short-term) reliability $R$ (i.e., repeatability) for a specific challenge $\mathbf{c}$ in the following way: Assume that the challenge $\mathbf{c}$ is evaluated $M$ times, and suppose the measured responses that are equal to 1 is $N$ out of $M$ evaluations. The reliability is defined as $R = N/M \in [0, 1]$.

In Eq. (1) we showed the mathematical relationship between the parity vector $\mathbf{\Phi}$ and the corresponding response $\varDelta$. Similarly, there exists a mathematical relationship between the reliability $R$ of a given challenge and its response $\varDelta$ as shown in [5]:

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

where the noise follows a normal distribution $\mathcal{N}(0, \sigma_N)$ and $\varPhi(\cdot)$ is the cumulative distribution function of the standard normal distribution. In the case where $R \in [0.1, 0.9]$ a further approximation [5] can be made:

$$\varDelta/\sigma_N \approx R. \tag{4}$$

### 2.3 The IPUF Design

A $(x, y)$-IPUF 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. We denote the input to the $x$-XOR APUF as $\mathbf{c}_x = (\mathbf{c}[0], \ldots, \mathbf{c}[i], \mathbf{c}[i + 1], \ldots, \mathbf{c}[n - 1])$. The response $r_x$ of the $x$-XOR APUF is interposed in $\mathbf{c}_x$ to create a new $n + 1$ bit challenge $\mathbf{c}_y = (\mathbf{c}[0], \ldots, \mathbf{c}[i], r_x, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$. The final response bit is the response

$r_y$ of the $y$-XOR APUF with respect to the new challenge $\mathbf{c}_y$. The structure of the $(x, y)$-IPUF is shown in Fig. 4.

When creating an $(x, y)$-IPUF three important parameters determine its security against machine learning attacks. The three parameters are the number of APUFs in the upper layer $x$, the number of APUFs in the lower layer $y$, and the place where the response of the upper layer $r_x$ is interposed in the input challenge for the $y$-XOR APUF. In the next two sections we go over more details related to the machine learning attacks on PUFs including the $(x, y)$-IPUF. Based on this knowledge, we explain how to properly choose the security parameters for the $(x, y)$-IPUF in Section 5 to secure it against various machine learning attacks.

## 3 Machine Learning Attacks

In this section we discuss three common machine learning attacks on PUFs without using physical side channel information such as power consumption, electromagnetic emission, etc. The two main attacks mentioned in the literature are classical machine learning attacks [26] which use CRPs and reliability based machine learning attacks which use challenge reliability pairs [5,3]. A hierarchy of the different attack types within classical and reliability based machine learning is shown in Fig. 2. We also introduce a third variation on these attacks specifically for use against the IPUF, the linear approximation (LA) attack. This attack can be used in combination with attack types 2, 3 and 5 in Fig. 2.

### 3.1 Classical Machine Learning Attacks

In classical machine learning, 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 [5,7,27,32,3,31]. In a classical machine learning attack, the relationship between the number of CRPs needed to model an $x$-XOR APUF and the number of APUFs in an $x$-XOR APUF is exponential. Therefore, when a large number of APUFs are used in an $x$-XOR APUF (high $x$) it becomes infeasible to model an $x$-XOR APUF [26]. When this occurs, we consider the classical machine learning attack to be mitigated or we say that the PUF is **secure** against classical attacks. According to [33], $x$ should be at least 10 to be secure against the state-of-art classical modeling attacks, i.e., derivative based modeling attack (Logistic Regression).

### 3.2 Reliability Based Machine Learning Attacks

Conceptually, instead of using CRPs like in the classical machine learning attack, the *repeatability* (i.e. short-term reliability) of APUF outputs can be used to build an APUF model based on a set of challenge reliability (not response) pairs. In this paper, we refer to such attacks as reliability based machine learning attacks. The relationship between reliability, $R$ and the weights $\mathbf{w}$, of an APUF were shown in Eq. (3) and Eq. (4). In [5] a reliability based machine learning attack was done under the assumption that $R \in [0.1, 0.9]$. Based on this assumption a system of linear equations was established using Eq. (4) so that $\mathbf{w}[i]/\sigma_N$ could be solved for using the *Least Mean Square* algorithm. However, reliability based machine learning attacks can be done **without** assumptions about the range of $R$, as we discuss next in Section 3.2.

**The CMA-ES Reliability Attack** In [2,3] a machine learning attack on APUFs was developed 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 $\epsilon$, 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. 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. (5) and Eq. (6), are found in [5] and [3], respectively:

$$R = \frac{1}{M}\sum_{i=1}^{M} r_i \tag{5}$$

$$R = |M/2 - \sum_{i=1}^{M} r_i| \tag{6}$$

Note similar to Eq. (5) and Eq. (6), repeatability for an APUF was defined as $R = N/M \in [0, 1]$ (see Section 2.2). The objective of CMA-ES is to learn weights $\mathbf{w} = (\mathbf{w}[0], \ldots, \mathbf{w}[n])$ together with a threshold value $\epsilon$. 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:

$$R'_i = \begin{cases} 1, & \text{if } |\Delta| \geq \epsilon \\ 0, & \text{if } |\Delta| < \epsilon, \end{cases} \tag{7}$$

   where $\epsilon = \epsilon_j$ and $\Delta$ follows from (1) and (2) with $\mathbf{c} = \mathbf{c}_i$ and $\mathbf{w} = \mathbf{w}_j$. Note that for a given model $\mathbf{w}_j$ with input $\mathbf{c}_i$, $R'_i$ indicates the reliability of the output response of the model.

(b) Compute the Pearson correlation $\rho_j$ based on the $(R_1, \ldots, R_i, \ldots, R_N)$ and $(R'_1, \ldots, R'_i, \ldots, R'_N)$, 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 $\rho$, and then, from these $L$ models, another $K$ models are generated based on the CMA-ES algorithm.
5. Repeat the steps (3)-(4) for $T$ iterations and model $(\mathbf{w}, \epsilon)$ which has highest Pearson correlation $\rho$ will be chosen as the desired model. Note that sometimes the chosen model may have low prediction accuracy and in this case we restart the algorithm to find a model with higher prediction accuracy.

While the above pseudo-code is used to model an APUF, it can also be used to model an $x$-XOR APUF. Let $\mathcal{Q} = \{(\mathbf{c}_1, R_1), \ldots, (\mathbf{c}_i, R_i), \ldots, (\mathbf{c}_N, R_N)\}$ be a set of challenge and reliability pairs of an $x$-XOR APUF instance. If the CMA-ES algorithm for modeling an APUF is executed many times with the set $\mathcal{Q}$, then it can produce $x$ different models for $A_0, \ldots, A_{x-1}$ with high probability [3]. 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 [3], one can parallelize CMA-ES executions to build models for $A_0, \ldots, A_{x-1}$.

As explained above, the modeling of an $x$-XOR APUF simplifies to the modeling of $x$ independent APUF instances in the CMA-ES reliability attack. This is significant because the relationship between $x$ and the number of CRPs needed for the CMA-ES reliability attack is linear. As previously stated, classical machine learning attacks have an exponential relationship between the number of CRPs and $x$. Therefore increasing $x$ is a valid way to mitigate the classical machine learning attack. However, the CMA-ES reliability attack cannot be defeated in the same manner.

**Enhancing the CMA-ES Reliability Attack** In this subsection, we briefly describe enhancements to the original CMA-ES reliability attack [3]. Using these alterations, we can model APUFs and XOR APUFs more accurately than the original attack proposed by Becker [3]. We experimentally demonstrate these results in Section 6.3 and Section 6.4.

In the CMA-ES reliability attack, each model $\boldsymbol{w}$ is evaluated according to a fitness function. The fitness function correlates the observed reliability $R$ of the PUF, to the reliability of the estimated model $R'$. Let us denote the observed reliability $R$ in Eq. (6) by $R_{original}$ and the model reliability $R'$ in Eq. (7) as $R'_{original}$. The correlation between $R_{original}$ and $R'_{original}$ is limited by the fact that the observed reliability $R_{original}$ only ranges from $[0, M/2]$, **while the**

**range of $R'_{original}$ is $\{0, 1\}$. It means, $R'_{original}$ does not capture $R_{original}$ very well.**

We propose two alternative definitions for $(R_{original}, R'_{original})$. We define $(R_{absolute}, R'_{absolute})$ and $(R_{cdf}, R'_{cdf})$, as follows:

$$R_{original} = |M/2 - \sum_{i=1}^{M} r_i| \quad \text{and} \quad R'_{original} = \begin{cases} 1, & \text{if } |\Delta| \geq \epsilon \\ 0, & \text{if } |\Delta| < \epsilon, \end{cases} \tag{8}$$

$$R_{absolute} = |M/2 - \sum_{i=1}^{M} r_i| \quad \text{and} \quad R'_{absolute} = |\Delta| \tag{9}$$

$$R_{cdf} = \frac{1}{M} \sum_{i=1}^{M} r_i \quad \text{and} \quad R'_{cdf} = \Phi(-\Delta/\sigma_N) \tag{10}$$

In both $(R_{absolute}, R'_{absolute})$ and $(R_{cdf}, R'_{cdf})$ the observed reliability and model reliability have the same range so that they exhibit better correlation than $(R_{original}, R'_{original})$. In addition $(R_{absolute}, R'_{absolute})$ only requires $n + 1$ parameters (the number of weights $\mathbf{w}$) to model an $n$ bit APUF. $(R_{cdf}, R'_{cdf})$ on the other hand requires the same number of parameters as $(R_{original}, R'_{original})$ because $\sigma_N$ must be estimated. However, the advantage of $(R_{cdf}, R'_{cdf})$ is that it retains the response information as well as reliability information by not using the absolute value operation when computing $R'_{cdf}$.

Using either $(R_{absolute}, R'_{absolute})$ or $(R_{cdf}, R'_{cdf})$ instead of $(R_{original}, R'_{original})$ improves the accuracy of the CMA-ES reliability attack. $(R_{cdf}, R'_{cdf})$ can create a more accurate model than $(R_{original}, R'_{original})$ when attacking an individual APUF. We hypothesize that this is because $(R_{cdf}, R'_{cdf})$ uses the proper reliability ranges and takes into account the response information of the APUF as well as the reliability information. Response information is taken into account in $(R_{cdf}, R'_{cdf})$ because the equation for $R'_{cdf}$ does not use the absolute value operation. However, when attacking an XOR APUF, the response of each individual APUF is not known so $(R_{cdf}, R'_{cdf})$ does not improve the CMA-ES attack in this case. In this case $(R_{absolute}, R'_{absolute})$ outperforms $(R_{original}, R'_{original})$ because it has the proper reliability ranges and does not take into account response information like $R'_{cdf}$. In the case of both APUF and XOR APUF designs, the original CMA-ES attack can be improved by using a more precise fitness functions. This is significant because due to the improved modeling the CMA-ES reliability attack can now work with less training data, where before the original attack would fail. When sufficient training data is available, the proposed fitness functions give more accurate models than the original fitness function. We experimentally verify our claims in Section 6.3.

### 3.3 The Linear Approximation Attack Against $(x, y)$-IPUFs

A third type of attack that can be used in conjunction with classical machine learning or reliability based machine learning is the linear approximation attack. This attack is specifically designed for use against the $(x, y)$-IPUF. We develop the attack based on the following observation: The difference between the input challenge to the $y$-XOR APUF in an $(x, y)$-IPUF and the input to a standard $y$-XOR APUF is only one bit. Assume $i + 1$ is the interposed bit position for the input challenge to the $y$-XOR APUF in an $(x, y)$-IPUF. The input to the $y$-XOR APUF is denoted as $\mathbf{c}_y = (\mathbf{c}[0], \ldots, \mathbf{c}[i], r_x, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$. Instead of attempting to learn the $x$-XOR APUF in the $(x, y)$-IPUF to estimate $r_x$, we can give a fixed value for bit $i+1$, i.e. $\mathbf{c} = (\mathbf{c}[0], \ldots, \mathbf{c}[i], 0, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$.

By making this approximation we can effectively ignore the $x$-XOR APUF component of the $(x, y)$-IPUF and treat the $(x, y)$-IPUF as a $y$-XOR APUF. Through this approximation we can do both classical and reliability based machine learning attacks on the $(x, y)$-IPUF while still using the XOR APUF model. In Section 5.2 we analyze under what conditions the linear approximation accurately approximates the $(x, y)$-IPUF and how this attack can be mitigated.

## 4 Analysis of the CMA-ES Reliability Attack

The CMA-ES reliability attack is a serious security issue when designing a PUF. Our goal is to create a secure PUF design (the $(x, y)$-IPUF) that defeats this attack. To do this, it is necessary to understand under what conditions the CMA-ES reliability attack works. In this section, we consider the following questions for in-depth analysis of the CMA-ES reliability attack:

1. In [3] it is noted that CMA-ES converges more often to some APUF instances than others when modeling the components of an $x$-XOR APUF. Essentially this means some APUFs in an $x$-XOR APUF are *easier* and some are *harder* to model using the given challenges reliability pairs. Why does this happen?
2. What are the conditions such that CMA-ES never converges to a particular APUF instance? In other words, under which condition does the CMA-ES reliability attack fail?

The first question was posed in [3] without any theoretical answer and the second question has not been investigated in the literature. The next experiments are aimed at answering these questions. Based on the knowledge gained from these experiments, we develop the $(x, y)$-IPUF that is secure against the CMA-ES reliability attack as well as the other machine learning attacks mentioned in Section 3.

### 4.1 Experiment-I: Understanding CMA-ES Convergence

The objective of this experiment is to analyze why some APUF instances are *easier* and some are *harder* to model using reliability based CMA-ES. More specifically, we want to verify whether the probability of converging to an APUF model in CMA-ES is correlated with the data set noise proportion of that APUF instance present in a given data set $\mathcal{Q}$. Here we define the data set noise proportion for a particular APUF in an $x$-XOR APUF as the number of noisy

| APUF model had the $n$th highest noise proportion in the data set | $n = 1$ | $n = 2$ | $n = 3$ | $n = 4$ | $n = 5$ | $n = 6$ | $n = 7$ | $n = 8$ | $n = 9$ | $n = 10$ | Failed to converge to any APUF model |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Percent of time convergence occurred | 36% | 5% | 9% | 5% | 12% | 7% | 7% | 5% | 7% | 5% | 2% |

Table 1: Relationship between the APUF model converged to by CMA-ES and that APUF's noise rate proportion in each $\mathcal{Q}$

(unreliable) CRPs due to the specified APUF divided by the number of total CRPs in $\mathcal{Q}$.

**Experimental Setup:** We run the CMA-ES algorithm 100 times on a 10-XOR APUF. Each time we run CMA-ES we use a new randomly generated dataset $\mathcal{Q}$ which contains $70 \times 10^3$ CRPs. The noise rate of each individual APUF is 20%. However it is important to note that while the noise rate of each APUF is the same, the data set noise proportion of each APUF in $\mathcal{Q}$ will change each time we generate a new $\mathcal{Q}$. For each $\mathcal{Q}$, we generate the challenge reliability pair for $\mathcal{Q}$ by measuring the response to each challenge 11 times.

**Experimental Results:** The results of experiment I are summarized in Table 1. In each run of CMA-ES, a model $\mathbf{w}$ is generated. We classify the model in the following way: if the model $\mathbf{w}$ matches one of the APUF models with probability $\geq 0.9$ or $\leq 0.1$ (complement model), then we accept it as a *correct* model for that particular APUF instance. If the generated model $\mathbf{w}$ does not match any of the APUF models we consider the CMA-ES algorithm to have failed to correctly converge. Note that $\mathbf{w}$ can match with at most one APUF instance. This is because if the model $\mathbf{w}$ corresponds to a particular APUF instance, then only that instance will have a matching probability $\geq 0.9$ or $\leq 0.1$, and the other APUF instances will have a matching probability around 0.5. This is due to the good uniqueness property of simulated APUF instances.

For the given $\mathcal{Q}$, we measure the data set noise proportion of each APUF with respect to the challenges present in $\mathcal{Q}$. If CMA-ES converges to the APUF model with the highest data set noise proportion in the current $\mathcal{Q}$, we increment the count in column one. If CMA-ES converges to the APUF model with the second highest data set noise proportion in the current $\mathcal{Q}$, we increment the count in column two and so on. If the CMA-ES algorithm generates a model that corresponds to none of the APUFs then we say it failed to converge to any model.

**Analysis of Results:** The experimental results can be explained as follows. Every time we generate a new set $\mathcal{Q}$, the challenge reliability pairs $\{(\mathbf{c}_i, R_i)\}$ of $\mathcal{Q}$ can be divided into two parts. The first part is made up of the reliable challenge reliability pairs $\mathcal{Q}_r$ and the second part is made up of the noisy challenge-reliability pairs $\mathcal{Q}_n$. Each APUF instance $A_i$ has its own set of noisy challenge-reliability pairs $\mathcal{Q}_{i,n} \subset \mathcal{Q}_n, i = 0, \ldots, 9$. The CMA-ES algorithm 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 reliability attack in Section 3.2). Since $\mathcal{Q}_r$ is useful for modeling all the 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, CMA-ES tries to converge to the APUF instance that has the largest value for $|\mathcal{Q}_{i,n}|/|\mathcal{Q}|$.

The PUF with the highest noise rate proportion in $\mathcal{Q}$ should have the highest Pearson correlation coefficient and therefore be the global maximum. However CMA-ES does not guarantee convergence to the global maximum. When CMA-ES converges to a local maximum, it produces another one of the valid APUF models, or an invalid model. For this reason we can see in Table 1 that we can converge to a PUF model that does not have the highest noise proportion with small probability compared to the highest case, i.e.,

$$\frac{5}{100}, \frac{7}{100}, \frac{9}{100}, \frac{12}{100} \ll \frac{36}{100}.$$

Overall every time we generate $\mathcal{Q}$, the PUF with the highest noise proportion in $\mathcal{Q}$ will have a high probability of being found by CMA-ES (i.e., 36/100). Likewise, the PUFs with a smaller noise proportion in $\mathcal{Q}$ will have a smaller probability of being found.

### 4.2 Experiment-II: CMA-ES Reliability Conditions

The objective of this experiment is to understand under which condition the CMA-ES attack fails to build a model for a particular APUF in an $x$-XOR APUF. From experiment I we know that CMA-ES is most likely to converge to the APUF model which has the highest data set noise proportion in $\mathcal{Q}$. We want to show that if the noise rate of the APUF instances in the $x$-XOR APUF's output are *not equal* (i.e., drawn from different distribution), then some APUF models cannot be generated by CMA-ES.

**Experimental Setup:** We simulate a 2-XOR APUF consisting of two APUF instances, denoted as $A_0$ and $A_1$. The noise rate of both APUFs are set at 1% (i.e., the reliabilities of each PUF are 99%). We run CMA-ES on the 2-XOR APUF 100 times. In each run of CMA-ES we generate a $\mathcal{Q}$ of size $70 \times 10^3$, where in $\mathcal{Q}$ each challenge is evaluated 11 times to generate the reliability information.

In this experiment we hypothesize that CMA-ES fails to build a certain APUF model when that model always has a lower noise rate (and therefore a lower data set noise proportion in $\mathcal{Q}$). To do this we manipulate the reliability information presented in the final output, such that $A_0$ always has lower data set noise proportion. This is achieved by applying majority voting to the responses

| $M^{\dagger}$ | Number of times $A_0$ found | Number of times $A_1$ found |
|---|---|---|
| 5 | 8 | 92 |
| 10 | 0 $^{\ddagger}$ | 99 $^{\ddagger}$ |

$^{\dagger}$ No. of measurements used in the majority voting of $A_0$.

$^{\ddagger}$ The sum of the two counts is not equal to 100 because one attack failed.

Table 2: Results of Experiment-II

of $A_0$ before XOR-ing it with the response of $A_1$. In majority voting, we have experimented with 5 and 10 votes to observe the performance of CMA-ES.

**Experimental Results:** The experimental results are shown in Table 2. The first column corresponds to the number of times the output was measured on $A_0$ before majority voting was done. The second and the third columns refer to the number of times the correct model was found by CMA-ES for $A_0$ and $A_1$, respectively.

**Analysis of Results:** From Table 2, it is clear that if $M$ is sufficiently large ($M \geq 10$), then the CMA-ES reliability attack cannot build a model for $A_0$. This is because we decreased the noise rate of $A_0$ by applying majority voting. Since $A_0$ is less noisy, it will have a lower data set noise proportion in $\mathcal{Q}$. As we established in the first experiment, CMA-ES tends to converge to the model with highest data set noise proportion with high probability. In Table 2 we can clearly see this happening when $M = 10$, as CMA-ES is never able to build a model for $A_0$ (the APUF with the lower data set noise proportion).

It is important to also note that just because the APUFs have different noise rates, it does not make the XOR APUF secure against the CMA-ES reliability attack. In the setup described in this experiment, an attacker could simply learn a model for $A_1$. Once the model for $A_1$ has been learned the attacker could then use that model to remove most of the noisy challenge-reliability pairs corresponding to $A_1$ from $\mathcal{Q}$. Doing this would create a new dataset $\mathcal{Q}_{reduced}$ from $\mathcal{Q}$ in which $A_0$ would have the highest data set noise proportion. The attacker could then run CMA-ES on $\mathcal{Q}_{reduced}$ to get a model for $A_0$.

### 4.3 Inferences from the Experiments

Two important points can be understood from the experiments in this section. In order for the CMA-ES reliability attack to be successful the following conditions must be met:

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

In the next section, we leverage the knowledge from these experiments to show how the proposed IPUF can be secured against the CMA-ES reliability attack.

# 5 Security and Reliability Analyses of $(x, y)$-IPUF

In this section, we analyze the security and reliability of the proposed $(x, y)$-IPUF design. There are three main parameters to choose when designing an $(x, y)$-IPUF. The three parameters are the number of APUFs in the upper layer $x$, the number of APUFs in the lower layer $y$ and the position of the interposed bit in the $y$-XOR APUF. By selecting the right parameter values, we show that our $(x, y)$-IPUF can be secured against all the aforementioned machine learning attacks enumerated in Section 3.

This section is organized as follows: To explain how to properly choose the $(x, y)$-IPUF parameters we first show how a single challenge bit in an APUF effects its output $r$, in Section 5.1. We then use this analysis to determine the most secure position for the interposed bit in a $(1, 1)$-IPUF in Section 5.2. While the $(1, 1)$-IPUF can defeat reliability based machine learning attacks, properly choosing the interposed bit position alone is not enough. The $(1, 1)$-IPUF still suffers from both classical and linear approximation machine learning attacks. Therefore, we expand our analysis to the $(x, y)$-IPUF where $x > 1$ and $y > 1$, in Section 5.3. By using a middle interposed bit and properly choosing $x$ and $y \geq 2$ we are able to show the $(x, y)$-IPUF is secure against all three types of machine learning attacks previously discussed. As an important result, we prove that derivative based classical modeling attack is not available to IPUF. Hence secure IPUF can have **more advantages in terms of security, reliability and hardware overhead** compared to secure XOR PUF against classical modeling attacks.

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

The influence of a challenge bit on an APUF's response depends on its position in the challenge $\mathbf{c}$ [23,6,24]. From Eq. (1), it can be observed that $\mathbf{\Phi}[j+1], \ldots, \mathbf{\Phi}[n-1]$ does not depend on the challenge bit $\mathbf{c}[j]$. For a given challenge $\mathbf{c}$, based on the linear delay model of the APUF, the delay difference $\mathit{\Delta}$ can be described as:

$$\mathit{\Delta} = (1 - 2\mathbf{c}[j]) \times \mathit{\Delta}_{Flipping} + \mathit{\Delta}_{Non-Flipping} \tag{11}$$

where $\mathit{\Delta}_{Flipping}$ is the term affected by the flipping of bit $\mathbf{c}[j]$ and is given by $\mathit{\Delta}_{Flipping} = \sum_{i=0}^{j} \mathbf{w}[i] \frac{\mathbf{\Phi}[i]}{(1-2\mathbf{c}[j])}$. Likewise, the term that is not dependent on the flipping of $\mathbf{c}[j]$ is denoted as $\mathit{\Delta}_{Non-Flipping} = \sum_{i=j+1}^{n} \mathbf{w}[i]\mathbf{\Phi}[i]$.

When $\mathbf{c}[j] = 0$ then $\mathit{\Delta} = \mathit{\Delta}_{Flipping} + \mathit{\Delta}_{Non-Flipping}$ and we can denote this $\mathit{\Delta}$ as $\mathit{\Delta}_{\mathbf{c}[j]=0}$ with corresponding response $r_{\mathbf{c}[j]=0}$. Similarly when $\mathbf{c}[j] = 1$ we will have $\mathit{\Delta} = -\mathit{\Delta}_{Flipping} + \mathit{\Delta}_{Non-Flipping}$. We denote this $\mathit{\Delta}$ as $\mathit{\Delta}_{\mathbf{c}[j]=1}$ and the response as $r_{\mathbf{c}[j]=1}$.

Essentially we want to know the influence of flipping $\mathbf{c}[j]$ on the output $r$. We measure this influence by computing the probability that the output remains the same if we flip the bit $\mathbf{c}[j]$ while keeping the rest of $\mathbf{c}$ constant:

$$\Pr_{\mathbf{c}}(r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1}). \tag{12}$$

Let us examine under which conditions $r_{\mathbf{c}[j]=0}$ will equal $r_{\mathbf{c}[j]=1}$. Assume for a specific challenge $\mathbf{c}$ we fix all bits (except for $\mathbf{c}[j]$) and the output $r$ is 0 regardless of the value of $\mathbf{c}[j]$. This means $\varDelta_{\mathbf{c}[j]=0} = \varDelta_{Flipping} + \varDelta_{Non-Flipping} > 0$ when $\mathbf{c}[j] = 0$ and $\varDelta_{\mathbf{c}[j]=1} = -\varDelta_{Flipping} + \varDelta_{Non-Flipping} > 0$ when $\mathbf{c}[j] = 1$. From this example, it is easy to derive that if and only if $r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1}$, then $|\varDelta_{Flipping}| \leq |\varDelta_{Non-Flipping}|$ so that:

$$\Pr_{\mathbf{c}}(r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1}) = \Pr(|\varDelta_{Flipping}| \leq |\varDelta_{Non-Flipping}|). \tag{13}$$

We follow existing PUF literature [17,16] and assume that when generating an instance of an APUF all $\mathbf{w}_i$ are sampled from the same normal distribution $\mathcal{N}(0, \sigma^2)$ and hence, $\varDelta_{Flipping} \sim \mathcal{N}(0, (j+1) \times \sigma^2)$ and $\varDelta_{Non-Flipping} \sim \mathcal{N}(0, (n-j) \times \sigma^2)$. Thus we have:

$$\begin{aligned}\Pr_{\mathbf{c}}(r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1}) &= \Pr(|\varDelta_{Flipping}| \leq |\varDelta_{Non-Flipping}|) \\ &= 4 \times \int_0^{\infty} \phi_{0,(n-j)\sigma^2}(u) \varPhi_{0,(j+1)\sigma^2}(-u)\, \mathrm{d}u,\end{aligned} \tag{14}$$

where $\phi_{\mu,\sigma^2}(\cdot)$ and $\varPhi_{\mu,\sigma^2}$ are the probability distribution function and cumulative distribution function of a normal distribution $\mathcal{N}(\mu, \sigma^2)$, respectively. Experimentally it has been shown [23,6,24] that Eq. (14) can be approximated as:

$$\Pr_{\mathbf{c}}(r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1}) \approx \frac{(n-j)}{n}, j = 0, \ldots, n-1 \tag{15}$$

This implies that the expected probability $\Pr_{\mathbf{c}}(r_{\mathbf{c}[j]=0} = r_{\mathbf{c}[j]=1})$ decreases with the increasing value of $j$, so the influence of each challenge bit is not equal. This undesirable security property means we must carefully consider the position for the interposed bit. In the next section we analyze how the interposed bit position effects the security and reliability of the $(1,1)$-IPUF.

### 5.2 Security and Reliability Analysis of the (1,1)-IPUF

The most basic form of the $(x, y)$-IPUF is the $(1,1)$-IPUF with the upper layer consisting of a single APUF $A_{upper}$ and the lower layer consists of a single APUF $A_{lower}$. Let us denote the responses to $A_{upper}$ and $A_{lower}$ by $r_{upper}$ and $r_{lower}$, respectively. The final output of the $(1,1)$-IPUF is the response $r_{lower}$. Based on the $(1,1)$-IPUF structure we analyze where to interpose the bit in the lower APUF and the $(1,1)$-IPUFs reliability and its security against machine learning attacks.

**Reliability of (1,1)-IPUF** In order to determine the effect of measurement noise in the $(1,1)$-IPUF, we evaluate a challenge $\mathbf{c}$ twice. The first time the challenge is applied let us denote $r_{upper,0}$ as the response of the upper APUF and $r_{lower,0}$ as the response of the lower APUF. Likewise, let us denote $r_{upper,1}$ and $r_{lower,1}$ as the APUFs responses the second time the challenge is evaluated.

Assume APUF $A_{upper}$ has noise rate $\beta_{upper}$ such that $\Pr_{\mathbf{c}}(r_{upper,0} \neq r_{upper,1}) = \beta_{upper}$. Similarly, assume that APUF $A_{lower}$ has noise rate $\beta_{lower}$ such that $\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1} | r_{upper,0} = r_{upper,1}) = \beta_{lower}$.

Let us denote $i+1$ as the interposed bit position for $r_{upper}$ in the $(n+1)$-bit challenge of $A_{lower}$. We derive $\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1}) =$

$$\begin{aligned} &\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1} | r_{upper,0} = r_{upper,1}) \Pr_{\mathbf{c}}(r_{upper,0} = r_{upper,1}) + \\ &\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1} | r_{upper,0} \neq r_{upper,1}) \Pr_{\mathbf{c}}(r_{upper,0} \neq r_{upper,1}) = \\ &\qquad\qquad \beta_{lower}(1-\beta_{upper}) + \left(\frac{i+1}{n+1}\right) \beta_{uppper} \end{aligned} \tag{16}$$

In practice $\beta_{upper} \ll 1$ and $\beta_{lower} \ll 1$, thus Eq. (16) can be approximated as:

$$\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1}) \approx \beta_{lower} + \beta_{upper} \frac{i}{n} \tag{17}$$

From Eq. (16) and Eq. (17) it can be seen that the reliability information of $A_{upper}$ and $A_{lower}$ available at the output of $(1,1)$-IPUF are not *equal* even when $\beta_{lower} = \beta_{upper}$. If we assume $\beta_{lower} = \beta_{upper} = \beta$ then $A_{lower}$ contributes approximately $\beta$ while $A_{upper}$ contributes approximately $\beta \frac{i}{n}$. The unequal reliability contribution shown by our analysis has important implications for the success of the CMA-ES reliability attack.

**Security of $(1,1)$-IPUF** We will now discuss the security of the (1,1)-IPUF with respect to the CMA-ES reliability attack and classical machine learning attacks. We also analyze the (1,1)-IPUFs resistance to the linear approximation attack.

***CMA-ES Reliability Attack*** The (1,1)-IPUF is theoretically secure against the CMA-ES reliability attack for two reasons when the interposed bit position for $r_{upper}$ is properly chosen. The first reason is based on the conditions under which the CMA-ES attack operates and the second is based on the computation done to learn the APUF model in CMA-ES.

First, recall the conditions under which the CMA-ES reliability attack works as described in Section 4.2. In order to successfully model the APUFs components, each APUF must contribute equal reliability information to the output. For the (1,1)-IPUF, we showed in Eq. (17) that the contributions of $A_{lower}$ and $A_{upper}$ are not equal, i.e., $\beta$ is not equal to $\beta \frac{i}{n}$ when $i$ is between 0 and $\frac{n}{2}$. Due to the unequal contribution of reliability information in the output when the interposed bit position $(i+1)$ is not close to $n$, CMA-ES will not converge to the model for $A_{upper}$. We did the following experiment to determine the importance of the interposed bit position. We launched the reliability based attack on a 64-bit (1,1)-IPUF with 30,000 CRPs. In this experiment, the noise each APUF was 20% ($\beta = 0.2$). The number of iterations for CMA-ES was 30,000. We repeated the attack 20 times with the interposed bit position at 0 ($i = 0$), 32 ($i = n/2$) and 64 ($i = n$). **The result shows that $A_{upper}$ can be modeled (i.e., the**

**prediction accuracy of the model of $A_{upper}$ is $98\%$) when $i = n = 64$. However, if the inserted position is in the middle $(32)$ or at first stage $(0)$, then $A_{upper}$ can not be modeled (i.e., the prediction accuracy of the model of $A_{upper}$ is $51\%$).**

Since we cannot first build a model for $A_{upper}$ we must first try to build a model for $A_{lower}$. This brings us to the second reason the (1,1)-IPUF is secure against the CMA-ES reliability attack. Recall in Section 3.2 that $\Delta$ is needed to compute the fitness of each model $\boldsymbol{w}$. $\Delta$ is based on the input to $A_{lower}$. However we do not know one of the input bits (the interposed bit) to $A_{lower}$ so we cannot compute $\Delta$ (see the calculation of $\Delta$ in Eqs. 1 and 2).

**However, we should take a closer look at the way the computation of $\Delta$ is done, to know how the interposed bit position affects the modeling attack on $A_{lower}$.** In Section 5.1, we have $\Delta = (1 - 2\mathbf{c}[j]) \times \Delta_{Flipping} + \Delta_{Non-Flipping}$ (see Equation 11) and thus, if $j$ gets closer to 0, then $\Delta$ and $\Delta_{Non-Flipping}$ become similar. Strictly speaking, we cannot run CMA-ES to build a model for $A_{lower}$ when the interposed bit position $i$ is **NOT** close to 0, for example the interposed bit position is in the range of $[n/2, n]$.

**Conclusion:** We cannot model $A_{upper}$ due to the unequal reliability information on the output and we cannot model $A_{lower}$ due to the unknown value of the interposed bit when the interposed bit is properly chosen. Therefore, we claim the $(1,1)$-IPUF is secure against the standard CMA-ES reliability attack when the interposed bit position is properly chosen in the middle of the input to $A_{lower}$.

***Classical Machine Learning Attacks*** The $(1,1)$-IPUF is not secure against classical machine learning attacks due to its low model complexity. Instead of modeling the APUF components individually, any machine learning algorithm can be used to learn the model for $A_{lower}$ and $A_{upper}$ simultaneously. Experiments to support our claim are given in Section 6 (see Table 3). Note that, we will prove that the derivative based classical modeling attacks is **not** available to IPUF in next section and thus, we only need to consider derivative free classical modeling attacks for IPUF.

***Linear Approximation Attack*** The security of the $(1,1)$-IPUF against the linear approximation attack depends on the choice of the interposed bit position. We introduced the linear approximation attack in Section 3.3. In this attack, any classical or reliability based machine learning attack on an XOR PUF can be adapted to work on an $(x,y)$-IPUF. This adaptation is done by approximating the $(x,y)$-IPUF as a $y$-XOR APUF by fixing the interposed bit from the $x$-XOR APUF to be 0. In the case of the $(1,1)$-IPUF this means that we ignore the component $A_{upper}$ and only model $A_{lower}$.

Let us denote $\boldsymbol{c}_{lower}$ as the input to the $A_{lower}$ and $\boldsymbol{c}'_{lower}$ to be the approximation of $\boldsymbol{c}_{lower}$ where we fix the interposed bit $r_{upper}$ in $\boldsymbol{c}_{lower}$ to be 0. We can write $L(\boldsymbol{c}'_{lower})$ as the output of the linear approximation and $A_{lower}(\boldsymbol{c}_{lower})$ as the output of the $(1,1)$-IPUF. We can now analyze how effective the linear approximation is. We measure the effectiveness of the approximation by computing the probability that the output of the linearized model $L(\boldsymbol{c}'_{lower})$ matches the output of $A_{lower}(\boldsymbol{c}_{lower})$:

$$p_{approx} = \Pr_{\mathbf{c}}(L(\mathbf{c}'_{lower}) = A_{lower}(\mathbf{c}_{lower})) \tag{18}$$

If $p_{approx}$ is high, then the $(1,1)$-IPUF **is accurately approximated** by APUF $L(\mathbf{c}'_{lower})$ and hence can be modeled using the linear approximation attack.

Let us assume the following conditions for the analysis of the attack and for the sake of explanation, we drop *lower* from $\mathbf{c}'_{lower}$ or $\mathbf{c}_{lower}$. We model a $(x,1)$-IPUF with APUF components that are 100% reliable, and the output $r_{upper}$ of the $x$-XOR APUF is uniform, i.e., $\Pr_{\mathbf{c}}(r_{upper} = 0) = \Pr_{\mathbf{c}}(r_{upper} = 1) = \frac{1}{2}$. Then,

$$\begin{aligned} p_{approx} = & \quad \Pr_{\mathbf{c}}(L(\mathbf{c}') = A_{lower}(\mathbf{c})) \\ & = \Pr_{\mathbf{c}}\left(L(\mathbf{c}') = A_{lower}(\mathbf{c}) | r_{upper} = 0\right) \Pr_{\mathbf{c}}(r_{upper} = 0) \\ & + \Pr_{\mathbf{c}}(L(\mathbf{c}') = A_{lower}(\mathbf{c}) | r_{upper} = 1) \Pr_{\mathbf{c}}(r_{upper} = 1) \\ & = \quad 1 \times 1/2 + \tfrac{n-i}{n} \times 1/2 = 1/2 + \tfrac{n-i}{2n}. \end{aligned} \tag{19}$$

Eq. (19) shows that $p_{approx}$ decreases as $i$ increases. **Note that our discussion holds for any $(x,1)$-IPUF and thus, it must be applicable to the $(1,1)$-IPUF**. We launched the reliability based linear approximation attack on a 64-bit (1,1)-IPUF. The number of CRPs was 30,000 and the noise rate was 20%. From the experiment, we learned that the prediction accuracy of the linear approximation attack on a 64-bit (1,1)-IPUF when $i = 0, 32, 64$ is equal to $97\%, 70\%$ and $51\%$, respectively. A similar result can be achieved by just applying the linear approximation without using reliability information.

A prediction accuracy of 50% is the worst a machine learning attack can do on a PUF with uniform binary output. Therefore, it would seem that picking the interposed bit positon to be as high as possible would result in the most secure $(x,y)$-IPUF design. However, below we will explain why choosing a high interposed position is not ideal.

***Interposed Bit Position*** In the $(1,1)$-IPUF the only design parameter we must choose is the interposed bit position. Fig. 5 shows the tradeoffs between choosing a high or low bit position. The higher the interposed bit position, the more influence $A_{upper}$ has on the $(1,1)$-IPUF. As a result the PUF model is more complex. It is more difficult to attack with classical machine learning attacks and the linear approximation attack is less accurate. However, a high bit position yields high noise on the output (less reliable). In addition, with a high interposed bit position and large enough number of input bits, the $(1,1)$-IPUF becomes equivalent to an XOR APUF in terms of susceptibility to reliability based machine learning attacks.

The conclusion from the analysis of the $(1,1)$-IPUF is that using the interposed bit position as the only security parameter is not enough to mitigate all

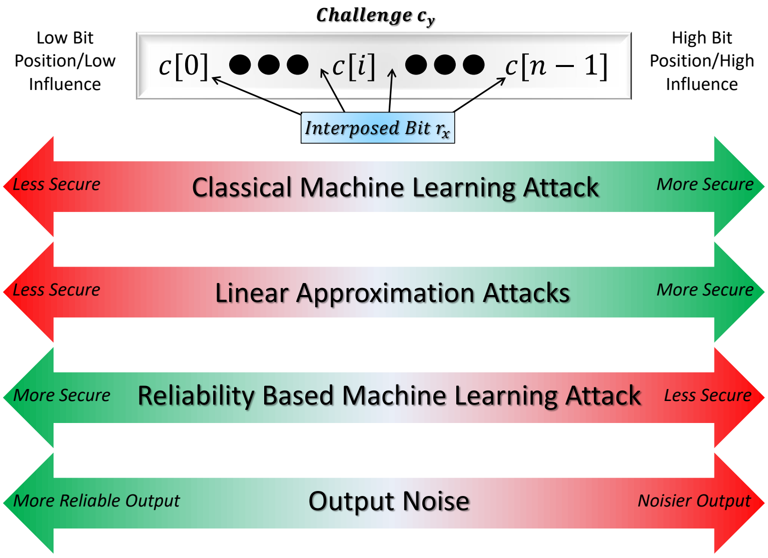

Challenge $c_y$
Low Bit Position/Low Influence
$c[0]$ ●●● $c[i]$ ●●● $c[n-1]$
High Bit Position/High Influence
Interposed Bit $r_x$
Less Secure
Classical Machine Learning Attack
More Secure
Less Secure
Linear Approximation Attacks
More Secure
More Secure
Reliability Based Machine Learning Attack
Less Secure
More Reliable Output
Output Noise
Noisier Output

Fig. 5: Relationship between the interposed bit position of $r_x$ and the security and reliability of the $(1, 1)$-IPUF

the different machine learning attacks. We seek a tradeoff between all the factors outlined in Fig 5 by choosing the interposed bit to be in the middle bit position. Based on this choice we then analyze how to modify $x$ and $y$ to further secure our design in Section 5.3.

### 5.3 Security and Reliability Analysis of the $(x, y)$-IPUF

The analysis in Section 5.2 showed that the $(1, 1)$-IPUF with a middle interposed bit position could defeat the CMA-ES reliability attack. However the $(1, 1)$-IPUF was still vulnerable to (derivative free) classical machine learning attacks and the linear approximation attack. In this section, we show how selecting $x$ and $y$ can mitigate the remaining machine learning attacks and we prove that the derivative based classical modeling attack is not available. We also analyze other important properties of the $(x, y)$-IPUF including reliability and the strict avalanche property.

**Security Analysis** In this section we show how the $(x, y)$-IPUF can mitigate the CMA-ES reliability attack, classical machine learning attacks and the linear approximation attack.

***CMA-ES Reliability Attack*** The security argument for the $(1, 1)$-IPUF in Section 5.2 also applies to the $(x, y)$-IPUF. Using challenge reliability pairs and

CMA-ES, an adversary cannot build models for the $x$ component APUFs in the $x$-XOR APUF. This is due to the unequal contribution of noise on the output between the $x$-XOR APUFs and the $y$-XOR APUFs. An adversary also cannot build models for the $y$ component APUFs in the $y$-XOR APUF as $r_x$ (the interposed bit position) cannot be predicted. Therefore, we consider the $(x, y)$-IPUF to be secure against the CMA-ES reliability attack.

***Classical Machine Learning Attacks*** There is no known way the APUF components of the $(x, y)$-IPUF can be modeled individually. As a result, the only way to attack an $(x, y)$-IPUF is by modeling all the $(x + y)$ component APUFs simultaneously using classical machine learning, e.g., neural network or CMA-ES based modeling attacks. Ruhrmair *et al.* [26] showed that the more APUFs that influence (contribute) to the output of an XOR APUF, the more difficult the PUF is to model using classical machine learning methods. In other words, increasing $x$ in an $x$-XOR APUF can mitigate classical machine learning attacks. Here we show that increasing $y$ in the $(x, y)$-IPUF can achieve a similar effect. First let us consider a classical machine learning attack on an $x$-XOR APUF. To accurately predict the output of an $x$-XOR APUF, a model for each individual APUF $A_i$, must be built. We must build a model for each APUF because each APUF contributes to the output. Here we define **contributes** in the following manner. If we flip the output of APUF $A_i$ (while the rest of the APUF outputs are held constant) and this causes the final response of the PUF $r$ to flip, then we say that $A_i$ **contributes** to the output. In an $x$-XOR APUF it can clearly be seen that each of the $x$ APUFs contribute to the output. This is because in an XOR gate flipping any one of the inputs always causes the output to flip. We know the more APUFs that contribute to the output, the harder the PUF design is to attack with classical machine learning. We also have defined what it means for an APUF component to contribute.

**Now we will analyze the $(x, y)$-IPUF to see how many APUFs contribute to the output** (essentially how difficult it is to attack with classical machine learning). We denote $r_y^0$ as the output of the IPUF when $r_x = 0$ with $\mathbf{c}_y = (\mathbf{c}[0], \ldots, \mathbf{c}[i], r_x = 0, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$ and $r_y^1$ as the output of the IPUF when $r_x = 1$ with $\mathbf{c}_y = (\mathbf{c}[0], \ldots, \mathbf{c}[i], r_x = 1, \mathbf{c}[i+1], \ldots, \mathbf{c}[n-1])$. Based on our definition of contribution above, if $r_y^0 = r_y^1$ it means that the $x$-XOR APUF output $r_x$ does not contribute to the final output $r_y$ of the IPUF. In this case only $y$ APUFs contribute to the output of the IPUF. Note we can also write $r_y^0 = r_y^1$ as $r_y^1 \oplus r_y^0 = 0$. Alternatively if $r_y^1 \oplus r_y^0 = 1$ then the output of $(x, y)$-IPUF depends on the output $r_x$ of $x$-XOR PUF, as well as the output $r_y$ of $y$-XOR PUF. This implies that there are $(x + y)$ APUFs which contribute to the final output $r_y$ for a given challenge. Therefore, the challenge-response space of an $(x, y)$-IPUF can be partitioned into two groups. The first group represents challenge-response pairs where the response only depends on the $y$ APUFs of $y$-XOR PUF. The second group of challenge-response pairs has responses which depend on both the $x$-XOR APUF and the $y$-XOR APUF (the response depends on a total of $(x + y)$ APUFs). Now we calculate the expected number of challenge-response pairs in each group. First, we will compute the probability the challenge-response pair is in the second group:

$$p_r = \Pr_{\mathbf{c}}(r_y^0 \neq r_y^1) = \Pr_{\mathbf{c}}(r_y^0 \oplus r_y^1 = 1) \tag{20}$$

Let $r_{lower,0}, r_{lower,1}, \ldots, r_{lower,y-1}$ be the outputs of the $y$ APUFs in the lower layer $y$-XOR PUF when $r_x$ is 0. We will assume ideal classical machine learning conditions where no measurement noise is present. Because there is no measurement noise Section 5.1 is applicable: For a given challenge $\mathbf{c}$, $r_{lower,i}$ depends on the upper layer output $r_x$ (in that output $r_{lower,i}$ would flip if $r_x$ would be substituted by 1) with probability $p = (i+1)/(n+1) \approx i/n$ if the feedback position of $r_x$ is $i$ (the probability is taken over the possible manufacturing variations which produce each of the component APUFs). For a given challenge $\mathbf{c}$, $p_r$ is the probability that the response of $(x, y)$-IPUF is equal to $r = 1 \oplus r_{lower,0} \oplus \ldots \oplus r_{lower,y-1}$ if $r_x$ would be substituted by 1. Then $p_r$ depends on $i$:

$$p_r = \sum_{k=1, k \text{ odd}}^{y} \binom{y}{k} p^k (1-p)^{y-k} = \frac{1-(1-2p)^y}{2}. \tag{21}$$

Thus, there is $1 - p_r$ of challenge-response pairs in the first group and $p_r$ of challenge-response pairs in the second group. For a given $(x, y)$-IPUF with parameter $i$, the expected number of APUFs contributing to the response of a given challenge is

$$\begin{aligned} &= (1-p_r)y + p_r(x+y) \qquad (22) \\ &= y + p_r x \end{aligned}$$

If $i = n$ and $y$ is odd, then $p_r = 1$ and $(x, y)$-IPUF is equivalent to $(x+y)$-XOR PUF in terms of security because of $(x+y)$ APUFs contributing to every challenge-response pairs. If $i = 0$, then $p_r = 0$ and $(x, y)$-IPUF is equivalent to $y$-XOR PUF. In terms of difficult of modeling with classical machine learning methods (i.e. CMA-ES), we can say that the $(x, y)$-IPUF with parameter $i$ is approximately equivalent to a $(y + p_r x)$-XOR PUF. We experimentally verify this claim in Section 6.1 (see Figs. 7 and 10).

To summarize this subsection, we showed that an $(x, y)$-IPUF is approximately as difficult to attack as an $(y + p_r x)$-XOR PUF when using classical machine learning methods. Therefore, we can use the same strategy employed in XOR APUF design [26] and increase $y$ and $x$ in an $(x, y)$-IPUF to mitigate classical machine learning attacks. It is also worth noting this analysis to determine the number of APUFs that contribute in an IPUF can be used to derive further IPUF properties such as uniformity, uniqueness and reliability.

***Linear Approximation Attack*** In the linear approximation attack we can approximate the $(x, y)$-IPUF by a $y$-XOR APUF, as discussed in Section 3.3.

The accuracy of the approximation for the $(x,1)$-IPUF was given in Eq. (19). Thus, if we use reliability based linear approximation attack on $(x,y)$-IPUF to model all $y$ APUFs at lower layer individually, the prediction accuracy of the approximated $y$-XOR APUF decreases when $y$ increases. For $y=1$ and the interposed bit in the middle position $p_{approx}=75\%$ (see Eq. 19). When $y=2$, the maximum prediction accuracy of the approximated model for the reliability based linear approximation attack is $p_{approx}=75\% \times 75\% + (1-75\%) \times (1-75\%) = 62.5\%$. The upper bound for the accuracy of this attack (as given in Eq. (19)) hold for $y \geq 2$, since introducing more APUF components in $y$-XOR APUF further decreases the prediction accuracy of the final model. This implies that for $y \geq 2$, $(x,y)$-IPUF is secure against the linear approximation attack based on reliability. We launched the CMA-ES based modeling attack on a 64-bit $(3,3)$-IPUF following this approach. In this experiment the noise for each APUF was 20%. The prediction accuracy of the final model was around 50% (since $y=3$ in this case) when 200,000 CRPs were used in the training phase (see Table 3).

Note that, we also can do the linear approximation with response information instead of reliability information. We approximate the $(x,y)$-IPUF by a $y$-XOR PUF. We treat the CRPs recorded by the $(x,y)$-IPUF as CRPs of a $y$-XOR PUF with the noise at least $100-p_{approx}$, or $\geq 25\%$ if the interposed bit in the middle. Compared to the reliability based linear approximation attack, this attack performs worse. This is due to the fact that classical machine learning attacks are sensitive to the noise and in this case, the noise is always larger than 25%. We did the CMA-ES classical machine learning attack (with the linear approximation) on a **reliable** 64-bit $(3,3)$-IPUF following this approach. The prediction accuracy of the model is around 50% when 200,000 CRPs are used in the training phase (see Table 3).

***Unavailability of Derivative based Modeling Attacks on IPUF*** In the analysis about the resistance of IPUF against classical modeling attacks, we **do not specifically** consider what type of classical modeling attacks here, i.e., black box methods, derivative based white box methods or derivative free white box methods. Now, we prove that the derivative based white box methods do not work for $(x,y)$-IPUF. This result is pretty important because **we can show that secure IPUFs against classical modeling attacks have more advantages in terms of security, reliability and hardware overhead compared to secure XOR PUF**.

As discussed in Section 1, the white-box machine learning techniques can be partitioned into two different categories: *derivative based modeling attacks* and *derivative free modeling attacks*. Basically, the former works more efficiently than the latter one when the searching space is large. For example, in [33], using Logistic Regression we can successfully model 4-XOR PUF with 15,000 CRPs only while using CMA-ES we cannot have a good model for 4-XOR PUF with 200,000 CRPs (see Figure 7). Hence, it is important to know if there exists any possible derivative based modeling attacks on IPUF or not. We show that the answer to this question is **NO**.

In the $x$-XOR PUF of IPUF, since there are $x$ $n$-bit APUF instances, we denote $\mathbf{w}^x = (\mathbf{w}_1^x, \dots, \mathbf{w}_x^x)$ as the model of $x$-XOR PUF and $\mathbf{w}_i^x = (\mathbf{w}_i^x[0], \dots, \mathbf{w}_i^x[n])$ are $(n+1)$ dimensional vectors and the models of APUFs in $x$-XOR PUF, $i = 1, \dots, x$. Similarly, $\mathbf{w}^y = (\mathbf{w}_1^y, \dots, \mathbf{w}_y^y)$ is the model the $y$-XOR PUF of IPUF and $\mathbf{w}_i^y$ are $(n+2)$ dimensional vectors and the models of APUFs in $y$-XOR PUF. In order to enable derivative based modeling attack, we follow the approach proposed in [26,29], i.e., we approximate the discrete output $r_x$ and $r_y$ by a continuous function sigmoid $\sigma(\cdot)$ where $\sigma(x) = \frac{1}{1+\exp(-x)}$. More precisely, we define the following functions:

$$\begin{aligned}
\mathit{\Delta}_x &= g_x(\mathbf{w}^x, \mathbf{c}) = \prod_{i=1}^{x} \mathbf{w}_i^x \mathbf{\Phi}(\mathbf{c}) \\
\hat{r}_x &= \delta(\mathit{\Delta}_x) + e(\mathit{\Delta}_x) = \delta(g_x(\mathbf{w}^x, \mathbf{c})) + e(g_x(\mathbf{w}^x, \mathbf{c})) \\
\mathit{\Delta}_y &= g_y(\mathbf{w}^y, \mathbf{c}, \hat{r}_x) = \prod_{i=1}^{y} \mathbf{w}_i^y \mathbf{\Phi}(\mathbf{c}, \hat{r}_x) \\
\hat{r}_y &= \sigma(\mathit{\Delta}_y) = \sigma(g_y(\mathbf{w}^x, \mathbf{c}, \hat{r}_y)),
\end{aligned}$$

where $\delta(x)$ is the stepsize function and $r_x = \delta(x)$, i.e., $\delta(x) = 0$ if $x > 0$, otherwise it is smaller than 1, and $e$ is a certain error function chosen by adversary. The function $\delta$ has derivative of 0 everywhere except $x = 0$ (i.e, derivative is of $\infty$) and $e(x)$ has derivative everywhere.

In order to find the optimal solution of $\mathbf{w} = (\mathbf{w}^x, \mathbf{w}^y)$ (i.e. the model for $(x, y)$-IPUF) from a randomly generated model $\mathbf{w}$, we define the following function as described in [26,29]:

$$l = -\frac{1}{N} \sum_{(\mathbf{c}_i, r_i), i=1,\dots,N} \ln(\sigma(\mathit{\Delta}_y)^{r_i}(1 - \sigma(\mathit{\Delta}_y))^{1-r_i})$$

where $\{(\mathbf{c}_1, r_1), \dots, (\mathbf{c}_N, r_N)\}$ are the challenge-response pairs of IPUF in training set. After that, we need to compute the gradient of $l$ in order to find the optimal solution, i.e., we need to compute

$$\begin{aligned}
\nabla l =& (\frac{\partial l}{\partial \mathbf{w}_1^x[0]}, \dots, \frac{\partial l}{\partial \mathbf{w}_1^x[n]}, \dots, \frac{\partial l}{\partial \mathbf{w}_x^x[0]}, \dots, \frac{\partial l}{\partial \mathbf{w}_x^x[n]}, \\
& \frac{\partial l}{\partial \mathbf{w}_1^y[0]}, \dots, \frac{\partial l}{\partial \mathbf{w}_1^y[n]}, \dots, \frac{\partial l}{\partial \mathbf{w}_y^y[0]}, \dots, \frac{\partial l}{\partial \mathbf{w}_y^y[n]})
\end{aligned}$$

After that we will update $\mathbf{w} = \mathbf{w} - \eta \nabla l$ where $\eta$ is a learning stepsize. By updating like this many times, we hope that the algorithm will converges to an optimal solution $\mathbf{w}^*$. Now, we focus on the calculation $\frac{\partial l}{\partial \mathbf{w}_i^x[j]}$. It can be computed as follows:

$$\begin{aligned}\frac{\partial l}{\partial \mathbf{w}_i^x[j]} &= \partial(-\frac{1}{N}\sum_{(\mathbf{c}_i,r_i),i=1,\ldots,N} \ln(\sigma(\Delta_y)^{r_i}(1-\sigma(\Delta_y))^{1-r_i}))/\partial \mathbf{w}_i^x[j] \\ &= -\frac{1}{N}\sum_{(\mathbf{c}_i,r_i),i=1,\ldots,N}[r_i(1-\sigma(\Delta_y))-(1-r_i)\sigma(\Delta_y)]\frac{\partial \Delta_y}{\partial \mathbf{w}_i^x[j]}\end{aligned}$$

We have

$$\begin{aligned}\frac{\partial \Delta_y}{\partial \mathbf{w}_i^x[j]} = \frac{\partial g_y}{\partial \mathbf{w}_i^x[j]} &= \frac{\partial g_y}{\partial[\delta(\Delta_x)+e(\Delta_x)]}\frac{\partial[\delta(\Delta_x)+e(\Delta_x)]}{\partial \mathbf{w}_i^x[j]} \\ &= \frac{\partial g_y}{\partial[\delta(\Delta_x)+e(\Delta_x)]}\frac{\partial e(\Delta_x)}{\partial \Delta_x}\frac{\partial \Delta_x}{\partial \mathbf{w}_i^x[j]}.\end{aligned}$$

But if we consider the linear approximation modeling attack where we fix $\hat{r}_x = 0 + e(\Delta_x)$, then we also have the same result, i.e.,

$$\frac{\partial \Delta_y}{\partial \mathbf{w}_i^x[j]} = \frac{\partial g_y}{\partial[\delta(\Delta_x)+e(\Delta_x)]}\frac{\partial e(\Delta_x)}{\partial \Delta_x}\frac{\partial \Delta_x}{\partial \mathbf{w}_i^x[j]}.$$

This fact implies that only we choose a correct error function, i.e., $e(\Delta_x) = r_x$ for every challenge $\mathbf{c}$ then the derivative based model works properly. If not, it will end up at the linear approximation modeling attack. It is obvious that we cannot build a correct error function $e(\Delta_x) = r_x$ for every challenge $\mathbf{c}$. The reason is that: if it is possible to do that, then it means we already successfully build the model of $x$-XOR PUF.

From this analysis, we conclude that derivative based modeling attack is equivalent to linear approximation modeling attack on IPUF which can be mitigated by choosing $y \geq 2$.

We already proved that $(x, y)$-IPUF with the interposed bit position at the middle is equivalent to $(y + \frac{x}{2})$-XOR PUF in terms of general security. From Inequality **??**, we can say that $(y + \frac{x}{2}) < 10$ to defeat derivative free modeling attacks on $(y + \frac{x}{2})$-XOR PUF. The number 10 is taken from [33], where the authors so that if $(y+\frac{x}{2}) = 10$, then the state-of-the art derivative based modeling attacks is infeasible to $(y + \frac{x}{2})$-XOR PUF. This means secure IPUFs against classical modeling attacks have more advantages in terms of security, reliability and hardware overhead compared to secure XOR PUF because secure IPUFs need less number of APUF instances than that of secure XOR PUF.

## Other $(x, y)$-IPUF Security Properties

***Reliability of the*** $(x, y)$***-IPUF*** By using an argument similar to the one used for the reliability of (1,1)-IPUF, we can write the noise of the $(x, y)$-IPUF as in Eq. (23):

$$\begin{aligned}\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1}) &= \beta_{lower}(1-\beta_{upper}) + \beta_{upper}(\frac{i+1}{n+1}) \\ &= \beta_{lower} - \beta_{lower}\beta_{upper} + \beta_{upper}(\frac{i+1}{n+1})\end{aligned} \tag{23}$$

where $\beta_{lower}$ is the noise rate of the lower $x$-XOR APUF, $\beta_{upper}$ is the noise rate of the upper $y$-XOR APUF, $i$ is the interposed bit position, $r_{lower,0}$ and $r_{lower,1}$ are the responses of lower layer of IPUF for the first and second evaluations when a challenge $\mathbf{c}$ is evaluated twice.

In practice $\beta_{lower}\beta_{upper}$ is much smaller compared to $\beta_{upper}$ and $\beta_{lower}$ and we use the following approximation $\frac{i+1}{n+1} \approx \frac{i}{n}$ to write the reliability of the $(x, y)$-IPUF as:

$$\Pr_{\mathbf{c}}(r_{lower,0} \neq r_{lower,1}) \approx \beta_{lower} + \beta_{upper} p_{bit} \tag{24}$$

where $p_{bit} = \frac{i}{n}$. The noise of the $(x, y)$-IPUF is equivalent to half that of an $(x + y)$ XOR APUF when $\beta_{lower} \ll \beta_{upper}$ and we pick the interposed bit to be the middle position, i.e. $y \ll x$ and $p_{bit} = 1/2$. In this case the $(x + y)$-XOR APUF has a noise rate of $\beta_{upper}$ and the $(x, y)$-IPUF has a noise rate of $\frac{\beta_{upper}}{2}$. Our claim is further verified by the experimental results presented in Section 6.2.

***Strict Avalanche Property of*** $(x, y)$***-IPUF*** The SAC property is an important security feature as discussed in [23,6,24]. Here we analyze the SAC property of $(x, y)$-IPUF. Assume that the output $r_x$ of $x$-XOR PUF is interposed at position $j$ in the challenge to $y$-XOR PUF, $j = 0, 1, \ldots, n$. We would like to compute the probability that flipping a bit in the input results in output bit of $(x, y)$-IPUF flipping. This analysis for the $(x, y)$-IPUF is similar to the analysis we did for the APUF in Section 5.1:

$$p_i = \Pr_{\mathbf{c}}(r_{c[i]=0} \neq r_{c[i]=1}), i = 0, 1, \ldots, n - 1 \tag{25}$$

where $r_{c[i]=0}$ and $r_{c[i]=1}$ are the output of $(x, y)$-IPUF when bit $\mathbf{c}[i] = 0$ and $\mathbf{c}[i] = 1$, respectively. To compute $p_i$, we consider the following two cases:

**Case I.** $\mathbf{c}[i]$ **flips,** $r_x$ **does not flip,** $r$ **flips**. In this case, we flip challenge bit $\mathbf{c}[i]$ but the output $r_x$ of $x$-XOR PUF does not flip and the $(x, y)$-IPUF's output $r$ flips (i.e., $r_{c[i]=0} \neq r_{c[i]=1}$). From the analysis in Section 5.1, we know that if the challenge bit $\mathbf{c}[i]$ flips, then the output of an APUF in an $x$-XOR PUF will flip with expected probability $p = \frac{i}{n}$ and thus, the output of $x$-XOR PUF will flip with an expected probability $p_x = \frac{1-(1-2p)^x}{2}$. In other words, $r_x$ will not flip with a probability $\bar{p}_x = 1 - p_x = \frac{1+(1-2p)^x}{2}$. When $r_x$ is not flipped due to $\mathbf{c}[i]$ flipping, the output of an APUF in the $y$-XOR PUF will be flipped by flipping $\mathbf{c}[i]$ with a probability $p' = \frac{i}{n+1}$. Thus the probability that $r_x$ does not flip (and $r$ flips) when flipping $\mathbf{c}[i]$ is equal to:

$$p_I = \frac{1 + (1 - 2p)^x}{2} \cdot \frac{1 - (1 - 2p')^y}{2}.$$

**Case II.** $\mathbf{c}[i]$ **flips,** $r_x$ **flips and** $r$ **flips**. If $r_x$ flips because $\mathbf{c}[i]$ flips, then there are two flipping challenge bits at position $i$ and $j$ in the input challenge to the $y$-XOR PUF. In this case, the output of an APUF in the $y$-XOR PUF will be flipped with a probability $p'' = \frac{|i-j|}{n+1}$. The computation for $p''$ is beyond the

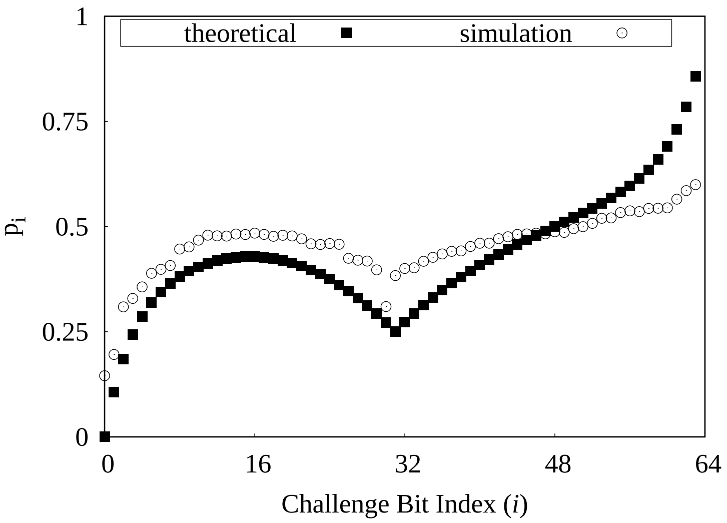


Fig. 6: SAC property of 64-bit (3,3)-IPUF with real and simulation data.

scope of this paper but a detailed derivation of it can be found in [24]. Thus the output $r$ will flip with a probability

$$p_{II} = \frac{1-(1-2p)^x}{2} \cdot \frac{1-(1-2p'')^y}{2}$$

Therefore, we have

$$p_i = p_I + p_{II} = \frac{1+(1-2p)^x}{2} \cdot \frac{1-(1-2p')^y}{2} + \frac{1-(1-2p)^x}{2} \cdot \frac{1-(1-2p'')^y}{2} \tag{26}$$

$$= \frac{1+(1-2\frac{i}{n})^x}{2} \cdot \frac{1-(1-2\frac{i}{n+1})^y}{2} + \frac{1-(1-2\frac{i}{n})^x}{2} \cdot \frac{1-(1-2\frac{|i-j|}{n})^y}{2}$$

We experimentally verify our calculations for the SAC property of the $(3,3)$-IPUF and the result is described in Fig 6. The simulation of the SAC property is computed by using 20,000 pairs of CRP for computing each $p_i$.

***Probably Approximately Correct (PAC) learning algorithm*** In [10,11,9], the authors proposed a novel modeling attack based on the PAC learning problem. As described in [11], assume that if the PUF has $k$ number of *influential bits* (see [11] for the detailed definition), then this PUF can be $\epsilon$-approximated by another Boolean function $h$ depending on only a constant number of Boolean variables $K$, where

$$K = e^{\frac{k}{\epsilon}\times(2+\sqrt{\frac{2\epsilon\log_2(4k/\epsilon)}{k}})} > e^{\frac{2k}{\epsilon}}.$$

As shown in [9], for a $n$-bit $(x, y)$-IPUF with interposed position $j$, $k$ can be computed as follows:

$$k = \sum_{i=0, i\neq j}^{n} p_i = \sum_{i=0, i\neq j}^{n} \{ \frac{1 + (1 - 2\frac{i}{n})^x}{2} \frac{1 - (1 - 2\frac{i}{n+1})^y}{2} + \frac{1 - (1 - 2\frac{i}{n})^x}{2} \frac{1 - (1 - 2\frac{|i-j|}{n})^y}{2} \}.$$

For 64-bit $(3, 3)$-IPUF with $j = 31$, $k = 25.2$. Hence, even if we consider a very weak approximated function $h$ with $\epsilon = 0.5$, then $K = e^{25.2\times 2/0.5} > e^{100} > 2^{100}$. It implies that it is impossible to launch the attack proposed in [9] on a 64-bit IPUF.

## 6 Simulation Results

We provide simulation results in this section to support the major security and reliability claims made regarding the IPUF. We also show the improvements our new models give to the original CMA-ES reliability attack on APUFs and XOR APUFs.

### 6.1 IPUF Security

**Simulated Machine Learning Attacks On IPUF and XOR APUF** To compare the vulnerabilities of the XOR APUF and various IPUF configurations to machine learning attacks we used the following setup: We simulated a 64-bit 6-XOR APUF, a 64-bit $(1, 1)$-IPUF and a 64-bit $(3, 3)$-IPUF using Matlab. Note that, all the IPUFs have the interposed bit in the middle position.

The APUF components that make up each PUF design use weights that follows a normal distribution with $\mu = 0$ and $\sigma = 0.05$. The noise for each weight follows a normal distribution $\mathcal{N}(0, \sigma^2_{\text{noise}})$. The distribution of each weight is therefore $\mathcal{N}(0, \sigma^2 + \sigma^2_{\text{noise}})$. In our simulations, we used the following relation between $\sigma$ and $\sigma_{\text{noise}}$ to control the reliability levels: $\sigma_{\text{noise}} = \gamma\sigma$, where $0 \leq \gamma \leq 1$. For $\gamma = 0$, a PUF instance is 100% reliable.

On each respective PUF design we run three different machine learning attacks (two in the case of the XOR APUF). We perform a classical machine learning attack (denoted as CML in Table 3 ), the CMA-ES reliability attack (denoted as RML in Table 3 ) and the linear approximation to the classical and reliability based machine learning attack (denoted as LA-CML and LA-RML in Table 3 ).

Each attack in Table 3 is performed using CMA-ES to optimize the mathematical model for 1000 iterations. Each attack uses 200,000 CRPs for training (or 200,000 challenge reliability pairs in the case of the reliability attacks). The accuracy of the attack is computed based on a testing dataset of 2000 CRPs.

| | | PUF Design | | |
|---|---|---|---|---|
| | | 6-XOR APUF | (1,1)-IPUF | (3,3)-IPUF |
| Attack | CML | 50.2% (✗) [†] | 82.4% (✓)[‡] | 52.9% (✗) |
| | RML | 84.0% (✓) | NA(✗) | NA (✗) |
| | LA-CML | NA(✗) | 74.0% (✓) | 49.0% (✗) |
| | LA-RML | NA(✗) | 73.0% (✓) | 48.0% (✗) |

[†] Attack fails
[‡] Attack works

Table 3: Vulnerability of different PUF designs to machine learning attacks. CML=Classical Machine Learning attack, RML=Reliability based Machine Learning attack, LA-RML=Linear Approximation Reliability based Machine Learning attack, LA-CML=Linear Approximation Classical Machine Learning attack.

We compute the accuracies reported in Table 3 by running each attack 10 times and taking the average.

The security of the 6-XOR APUF is shown in Table 3 in the first column. It is clear that an XOR APUF using a large number of APUF components (in this case 6) can mitigate a classical machine learning attack, given the amount of training data provided in this setup. However, the 6-XOR APUF is still highly vulnerable to reliability based machine learning attacks. This result is demonstrated in Table 3 as the CMA-ES reliability attack achieves an average accuracy of 84% on this PUF design. Note that we do not run the linear approximation attack on the 6-XOR APUF because this attack is specific to the IPUF.

The resilience of the $(1,1)$-IPUF to the three types of machine learning attacks is shown in column two of Table 3. We note the following: As hypothesized, the $(1,1)$-IPUF is vulnerable to classical machine learning due to its low model complexity. However, the linear approximation CMA-ES reliability attack (LA-RML) works on the $(1,1)$-IPUF which may seem unexpected given our previous claims. Recall that in general the CMA-ES reliability attack cannot be performed on any IPUF design (hence the X in entry for the RML attack for the $(1,1)$-IPUF). This is because the input to the $y$-XOR APUF is unknown and therefore $\varDelta$ cannot be computed. However, in our LA-RML attack on the $(1,1)$-IPUF and $(3,3)$-IPUF we do not use the original formulation of the CMA-ES attack. To make the attack work on IPUF configurations we assume the interposed bit to be 0. By doing this we can now compute $\varDelta$ and treat the $(1,1)$-IPUF as a single APUF. In section 5.3 we showed that for the $(x,1)$-IPUF using any machine learning technique with the linear approximation would have an upper bounded model accuracy of 75%. The model accuracy produced by the LA-RML and LA-CML on the $(1,1)$-IPUF in Table 3 closely matches our theoretical upper bound.

The security of the $(3,3)$-IPUF against each machine learning attack is demonstrated in the third column of Table 3. It can clearly be seen that all three attacks fail to produce an accurate model of the $(3,3)$-IPUF. Due to the

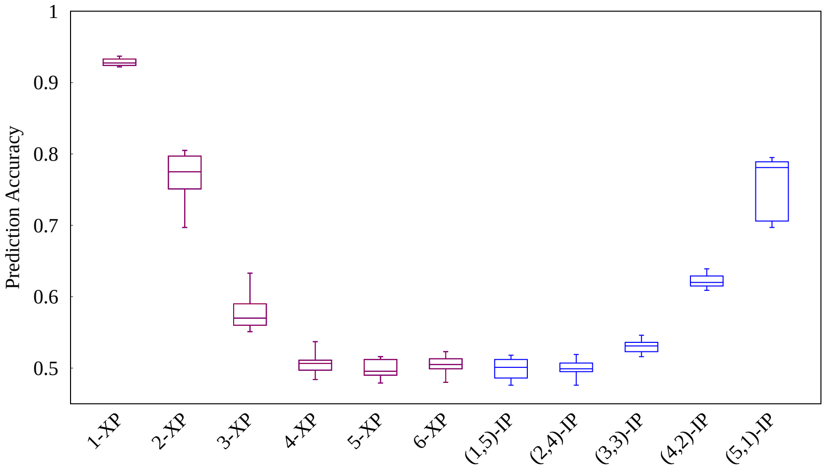
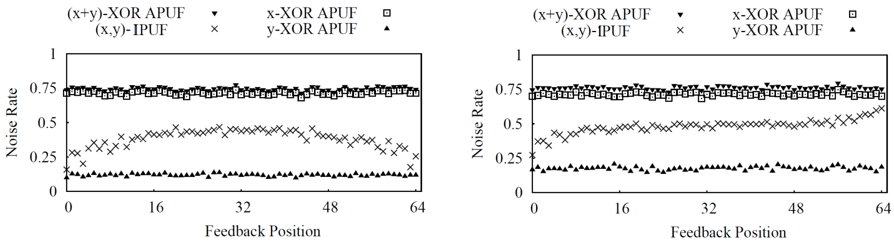


Fig. 7: Prediction accuracy of CMA-ES based modeling attack on 64-bit 2,3,4,5,6-XOR PUF (y-XP) and (1,5), (2,4), (3,3), (4,2), (5,1)-IPUF ((x,y)-IP).

high model complexity, the $(3, 3)$-IPUF is not vulnerable to the classic machine learning attack, just like the 6-XOR APUF. Just like for the $(1, 1)$-IPUF, the CMA-ES reliability attack cannot be performed on the $(3, 3)$-IPUF. When we run the linear approximation CMA-ES reliability attack, due to the higher $y$, the accuracy of the model generated by any method that uses the linear approximation is theoretically upper bounded at 75% (See Eq. (19)). This coincides with the low accuracy of the model generated by both the linear approximation CMA-ES reliability attack and the linear approximation classical machine learning attack. Overall our experimental results for this section support our claim that the $(x, y)$-IPUF with proper parameter choices is secure against all three types of machine learning attacks.

**$(x, y)$-IPUF and $(x + y)$-XOR APUF Model Complexity Comparisons** In Section 5.3, we compared the $(x, y)$-IPUF and the $(x + y)$-XOR PUF. Since we only focus on the design, we consider reliable PUFs to experimentally demonstrate Eq. (22). If the interposed bit position is in the middle, then the $(x, y)$-IPUF is equivalent to a $(y + \frac{x}{2})$-XOR PUF. We run the CMA-ES reliability attack on 64-bit APUF (a.k.a 1-XOR PUF or 1-XP), 64-bit 2,3,4,5,6-XOR PUF (y-XP) and (1,5), (2,4), (3,3), (4,2), (5,1)-IPUF ((x,y)-IP) with 200,000 CRPs for training. In each attack the CMA-ES algorithm is run for 1000 iterations. The results are shown in Fig 7. For each PUF, we attack it 10 times. The prediction accuracy of each attack is computed using 2000 CRPs. Fig 7 shows that the experimental results closely matches the theory presented in Eq. (22).



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

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

Fig. 8: Noise rate of (x+y)-XOR APUF, (x,y)-IPUF, x-XOR APUF and y-XOR APUF when the noise rate of each APUF is around 0.05.

### 6.2 IPUF Reliability

To compare the reliability of the $(x, y)$-IPUF to the $(x + y)$-XOR APUF we simulated a $(x + y)$-XOR APUF, $(x, y)$-IPUF, $x$-XOR APUF and a $y$-XOR APUF for $x = 20$, and $y = 2$ and $y = 3$. In the simulation, we have 64-bit APUFs, each with a noise rate defined by setting $\sigma_{\text{noise}} = 0.05\sigma$. To estimate the reliability, we evaluated each PUF design with 10,000 randomly generated challenges. 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 interpose position $i$ of the IPUF from 0 to 64. For each interpose position we measured the reliability of the $(x, y)$-IPUF. In the same figure we also plot the reliability of the $(x + y)$-XOR APUF, $x$-XOR APUF and $y$-XOR APUF. The simulated results are presented in Fig. 8.

Figures 8a and 8b show that the noise rate of the $(x+y)$-XOR APUF and the $x$-XOR APUF are close to each other as the value of $y$ is very small compared to the value of $x$ (i.e. $y = 2$ or $y = 3$). The noise rate of the $y$-XOR APUF is very small compared to the $(x + y)$-XOR APUF and the $x$-XOR APUF. The noise rate of the $(x, 3)$-IPUF increases when the interpose bit position increases from 0 to 64. However, the noise rate of the $(x, 2)$-IPUF reaches the maximum value at interpose position 32. This is due to the parity of $y$ (see Eq. (21)). The noise rate of the $(x, 3)$-IPUF is equal to half of the noise rate of the $(x+3)$-XOR APUF when $i$ is in the middle position. The experiments confirm our findings related to reliability (see Section 5).

### 6.3 Enhanced Reliability Based CMA-ES Attack On APUF

Table 4 depicts the modeling accuracy of the CMA-ES reliability attack on a 64-bit APUF. In this set of simulations, we have followed the same setup mentioned in the previous sections with $\sigma_{\text{noise}} = \sigma/10$. For this value of $\sigma_{\text{noise}}$, the reliability of the APUF is around $96 - 97\%$. From Table 4, it is evident that

| $N^{\dagger}$ | **Modeling Accuracy(%)** | | |
|---|---|---|---|
| | $(R_{original}, R'_{original})$ | $(R_{absolute}, R'_{absolute})$ | $(R_{cdf}, R'_{cdf})$ |
| 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.

Table 4: A comparison of 64-bit APUF's modeling accuracy using CMA-ES

| **Model setup** | $N^{\dagger}$ | **Modeling Acc.(%)** | | | | **Frequency**$^{\star}$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $A_0$ | $A_1$ | $A_2$ | $A_3$ | $A_0$ | $A_1$ | $A_2$ | $A_3$ |
| $(R_{o.}, R'_{o.})$ | $10 \times 10^3$ | 65.10 | 66.17 | 67.40 | 68.96 | 0 | 0 | 0 | 0 |
| | $20 \times 10^3$ | 98.08 | 97.91 | 98.23 | 98.33 | 1 | 4 | 3 | 1 |
| | $30 \times 10^3$ | 98.27 | 98.06 | 98.27 | 98.39 | 19 | 10 | 6 | 3 |
| | $50 \times 10^3$ | 98.31 | 98.16 | 98.37 | 98.44 | 39 | 20 | 17 | 10 |
| $(R_{a.}, R'_{a.})$ | $10 \times 10^3$ | 97.53 | 97.09 | 97.10 | 95.24 | 22 | 24 | 31 | 8 |
| | $20 \times 10^3$ | 97.86 | 97.75 | 97.74 | 97.78 | 24 | 29 | 29 | 17 |
| | $30 \times 10^3$ | 98.08 | 97.89 | 98.02 | 98.12 | 47 | 27 | 20 | 6 |
| | $50 \times 10^3$ | 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.

Table 5: Modeling results of 4-XOR APUF using CMA-ES and different reliability models

the CMA-ES reliability attack using our proposed model $(R_{absolute}, R'_{absolute})$ and $(R_{cdf}, R'_{cdf})$ outperforms the CMA-ES reliability attack using the original model $(R_{original}, R'_{original})$ as proposed in [3]. In addition, $(R_{cdf}, R'_{cdf})$ outperforms $(R_{absolute}, R'_{absolute})$ as both the response polarity and reliability information are considered in $(R_{cdf}, R'_{cdf})$. Although $(R_{cdf}, R'_{cdf})$ is better than $(R_{absolute}, R'_{absolute})$ for modeling a single APUF, in the context of an XOR APUF, we have observed from the experimental results that the performance of $(R_{original}, R'_{original})$ and $(R_{absolute}, R'_{absolute})$ are superior to $(R_{cdf}, R'_{cdf})$. One reason is that considering the response information in $R_{cdf}$ makes the modeling task more difficult in the context of an XOR APUF. Next, we discuss the performance of $(R_{original}, R'_{original})$ and $(R_{absolute}, R'_{absolute})$ in the context of XOR APUF modeling.

### 6.4 Enhanced Reliability Based CMA-ES Attack On XOR APUF

In Section 4, we discussed XOR APUF modeling using CMA-ES and the model $(R_{original}, R'_{original})$. Using our proposed model $(R_{absolute}, R'_{absolute})$ we can achieved better modeling accuracy for XOR APUF using less number of $(\mathbf{c}, R)$ pairs compared to using CMA-ES with the original model $(R_{original}, R'_{original})$.

To demonstrate the performance of the model $(R_{absolute}, R'_{absolute})$, we simulated a 4-XOR APUF in Matlab. We followed the same simulation setup mentioned in Section 6 with $\sigma_{\text{noise}} = \sigma/10$. For this value of $\sigma_{\text{noise}}$, the reliability of APUF instances are around $96 - 97\%$. Table 5 shows the performance comparison of $(R_{original}, R'_{original})$ and $(R_{absolute}, R'_{absolute})$ when modeling a 4-XOR APUF. In Table 5, we have reported two aspects: i) modeling accuracy and ii) frequency of the correct APUF models (a correct model has prediction accuracy greater than 90%). For the attack we run CMA-ES 100 times. It can be observed that $(R_{absolute}, R'_{absolute})$ results in more successful models compared to $(R_{original}, R'_{original})$. Note that in the case where $N = 10 \times 10^3$, no successful APUF model is built if $(R_{original}, R'_{original})$ is used. However using $(R_{absolute}, R'_{absolute})$, models of the APUFs in the XOR APUF can be successfully built in this case. The modification of the model $(R_{original}, R'_{original})$ to get $(R_{absolute}, R'_{absolute})$ results in a more efficient modeling of an XOR APUF.

# 7 IPUF Implementation

In order to validate our IPUF security and reliability claims, we implemented our proposed IPUF designs on an FPGA board. In this section, we describe the FPGA implementation details and related experimental results. We also discuss the limitations of FPGA based APUFs on composite PUF (XOR PUF and IPUF) security.

## 7.1 IPUF Implementation Details

In our IPUF design we we implemented every stage (switch) of each APUF by a Look-Up Table (LUT)[1]. The LUTs are chained together and the final output is collected by a flip-flop serving as an arbiter. The main issue in implementing any IPUF design is to make sure that the response $r_{upper}$ of the upper layer XOR APUF is ready when the interposed position of the lower layer XOR APUF is evaluated. To solve this issue, we added one more signal from the upper layer XOR APUF to inform the control circuitry when $r_{upper}$ is ready. Once the signal is received the lower layer XOR APUF evaluates its input.

Desirable statistical properties of a PUF FPGA implementation include uniqueness, reliability and uniformity. It has been experimentally verified that uniqueness is a serious issue when implementing FPGA based APUFs [21,13]. The main reason implementing unique APUFs on FPGAs is problematic [19,21] is because the designers are not allowed to precisely control the routing between each LUT to maintain the balance of the length of the two competing paths, the delay difference induced by routing is much larger than the delay difference

[1] In particular on our board, each stage is implemented by a 6-input 2-output LUT, where the two outputs serve as the outputs of upper and lower paths and only three inputs are used as the inputs of upper and lower paths and the challenge bit.

introduced by process variation. Thus, the behavior of one APUF on an FPGA is largely determined by the place and route of the LUTs.

In order to keep the balance between the delays of two competing paths, usually one switch chain is kept in one column of slices on an FPGA. However, in our APUF implementation on a Digilent Nexys 4 DDR with Xilinx Artix-7 embedded [8] there are four LUTs in each slice of the FPGA. Due to the configuration on this hardware, a choice must be made on how to connect the LUTs to form the switch chain. Since the behavior of each APUF is dominated by the routing difference between the two paths, each switch chain must be placed differently for each APUF, in order to create unique APUFs. Note this design strategy only alleviates the uniqueness issue on a single FPGA. If the same bitstream is used to program different FPGAs, the difference of the same APUFs on different FPGAs will be very small. Thus, this is not a general design strategy to improve the uniqueness of APUFs on FPGA, but it is sufficient for us to conduct our experiments.

We have two different ways to place the switch chains: (1) Random placement: we randomly select one LUT in each slice and then connect them together. (2) Pattern placement: we place the switches according to a pre-defined pattern. For example, we only use LUT A and LUT B in every slice. According to our experiments, each design strategy has advantages and disadvantages. For the random placement design strategy, the advantage of this method is that it gives us many options to build unique APUFs. Here we define two APUFs as being non-unique when their responses are the same for more than 60% of the challenges. Initially, we generated 100 different switch chains using random placement. After extensive evaluation, we selected 23 placements of the APUFs, which can provide APUFs with good uniformity (50.2% - 61.6%) and good uniqueness/inter-hamming distance (39.5% - 59.0%). The noise rate of these APUFs under room temperature is between 0.66% and 1.25%. However, with good statistic properties, comes a security weakness in the APUFs. Since the routing between each adjacent switches is randomly selected, there are a few delays that are significantly larger than the other delays. This effectively introduces a few significant weights in the feature vector of this APUF. As a result the difficulty of all machine learning modeling attacks is reduced since only these significant weights need to be precisely modeled (instead of having to precisely learn all the weights). We built the model of individual APUFs to understand the distribution of weights. The standard deviation of the weights of the APUFs created by random placement is from 4.23 to 7.48. As we will explain shortly, the standard deviation is much smaller for the pattern placement design strategy, but this method has its own drawbacks.

For the pattern placement design strategy, there are a very limited number of patterns we can generate and some APUFs formed by different placement patterns are not unique. After exhaustively trying 11 patterns[2], we found only 4 placement patterns that can generate 4 unique APUFs. The uniqueness limita-

[2] 1 pattern that uses all four LUTs in each slices, 6 patterns that uses a combination of two LUTs in each slices, 4 patterns that uses one LUTs in each slices.

tion of this design strategy gives us very few options to form an IPUF design with more than 4 APUFs in total. However, the pattern placement design strategy does not have the same security weakness as the random placement design strategy. We also built the model of individual APUFs to understand the distribution of weights in this design. The standard deviation of the weights of the APUFs created by pattern placement is from 1.11 to 3.77. Using this design strategy we do not observe the same effect of only a few influential weights (unlike in the random placement design strategy).

Each method has its own strengths and weakness. Since the design strategy will largely influence the experimental results that we will present later, we will clearly state how the APUFs are generated for each experiment.

### 7.2 Experimental Results

***Reliability Based Machine Learning Attack on XOR APUFs*** First, we repeated the enhanced CMA-ES reliability attack in Section 6.4 on XOR APUFs to validate the effectiveness of our attack. In this experiment, we selected 6 unique APUFs created by random placement to form a 6-XOR APUF on the FPGA. We then measured 300,000 CRPs with each CRP measurement repeated 11 times to get 300,000 challenge reliability pairs. The number of challenge reliability pairs used for one CMA-ES attack and the modeling results after 100 runs of CMA-ES are presented in Table 6.

| | #CRPs used in one attack | Overall Noise Rate | Average Prediction Accuracy |
|---|---|---|---|
| 2-XOR | 50,000 | 1.44% | 98.22% |
| 3-XOR | 90,000 | 2.38% | 96.38% |
| 4-XOR | 140,000 | 2.92% | 96.15% |
| 5-XOR | 200,000 | 3.80% | 96.62% |
| 6-XOR | 260,000 | 4.47% | 91.58%* |

Table 6: Results of reliability based machine learning attack on XOR APUFs with real measurements from the FPGA. *Note that, the attack on 6-XOR APUF only recovered 5 out of 6 models. The attacks on 2,3,4,5 XOR APUFs successfully recovered all the models.

***Reliability Based Machine Learning Attack on IPUFs*** To show the modeling resistance of the IPUF to the enhanced CMA-ES reliability attack, we perform the attack on a (1,1)-IPUF. We do not test this attack on IPUF designs with more APUF components because our security against reliability based machine learning attacks does not depend on the number of APUFs components.

First, we used two APUFs created by random placement to form a (1,1)-IPUF with an interposed position exactly in the middle in the lower APUF. We measured 200,000 CRPs with each with each CRP measurement repeated 11 times to get the challenge reliability measurements. We then sampled a subset of 90,000 challenge reliability pairs out of 200,000 challenge reliability pairs to run the CMA-ES reliability attack. CMA-ES was able to converge to the model of the upper APUF with 96.6% accuracy. According to our previous analysis and simulation results, only the model of the lower APUF should be built with up to 75% accuracy (i.e., the reliability based machine learning attack on a (1,1)-IPUF is equivalent to the reliability based linear approximation attack). When CMA-ES converged to the model of the lower APUF, the accuracy was upper bounded by 75%, which confirms our theoretical analysis.

We also tested the pattern placement design strategy. We used two APUFs created by pattern placement to form another (1,1)-IPUF with an interposed position exactly in the middle in the lower APUF. Again, we repeated the experiment above. In this case CMA-ES was not able to converge to the upper APUF after 10 runs. This validated our security claim that IPUF structure can prevent reliability based machine learning attacks. Unfortunately, due to the uniqueness issue of pattern placed APUFs, we were not able to test the security of the (3,3)-IPUF as we did in our experimental simulations.

***Analysis of Biased Weights vs Balanced Weights in Simulation*** When we discovered the phenomenon of biased weights in the APUFs generated by random placement, we further investigated why this implementation affects the security of IPUFs. We created APUF models with biased weights, such that the distribution that generates the large weights is $\mathcal{N}(0, 6)$, while the standard weights are drawn from $\mathcal{N}(0, 1)$. We also precisely controlled the number of large weights in the second half (the half which is closer to the output) of the lower APUF. We simulated (1,1)-IPUFs with the 4, 8, 12, 16, and 20 dominating weights. On each type of IPUF we performed the enhanced CMA-ES reliability attack 10 times. The results are shown in Table 7.

| | SAC | Upper | Lower | Failed |
|---|---|---|---|---|
| 4 | 0.39 | 0 | 10 | 0 |
| 8 | 0.58 | 9 | 1 | 0 |
| 12 | 0.54 | 2 | 8 | 0 |
| 16 | 0.32 | 0 | 9 | 1 |
| 20 | 0.41 | 0 | 10 | 0 |

Table 7: Results of the enhanced CMA-ES reliability attack on (1,1)-IPUF with biased weights.

The SAC value is computed by the probability that a bit flip in the middle challenge bit of the lower APUF will flip the final output bit. The larger the SAC value is, the higher the chance that the reliability information of the upper APUF will be exposed to adversaries. This leads to a successful attack on the upper APUF with higher probability. Ideally, the SAC value of the middle bit should be 50%, but due to the biased distribution in the weights on FPGAs, the SAC value can be possibly higher than normal. From the computed SAC values, the CMA-ES reliability attack **is supposed to not** converge to the **upper** APUF because they are good enough to prevent the attack according to our analysis in Section 5.2. However, the attack still works very well when the number of dominating weights is 8 or 12. We can explain this fact as follows. The interposed bit at the middle splits the lower APUF into two parts: the flipping part which is closer to the 0-th challenge bit and the non-flipping part which is closer to the output of IPUF. For the sake of explanation, we assume that all small weights are equal to 0. The motivation of assuming this is that if the biased weights are significant larger than the small weights, we can ignore all the small weights. We assume that there are $k$ biased weights in the flipping part and $m$ biased weights in the non-flipping part. Moreover, the biased weights follow a normal distribution $\mathcal{N}(0, \sigma_b^2)$. From Equation (11), we can write

$$\Delta = (1 - 2\mathbf{c}[n/2]) \times \Delta_{Flipping} + \Delta_{Non-Flipping}$$

where $\Delta_{Flipping} = \sum_{i=0}^{n/2} \mathbf{w}[i] \frac{\mathbf{\Phi}[i]}{(1-2\mathbf{c}[n/2])}$ and $\Delta_{Non-Flipping} = \sum_{i=n/2+1}^{n} \mathbf{w}[i]\mathbf{\Phi}[i]$. Since we have $k + m$ biased weights, $\Delta_{Flipping}$ and $\Delta_{Non-Flipping}$ only depend on $k$ and $m$ weights, respectively. This implies $\Delta_{Flipping}$ and $\Delta_{Non-Flipping}$ follow normal distributions $\mathcal{N}(0, k\sigma_b^2)$ and $\mathcal{N}(0, m\sigma_b^2)$, respectively. We know that, for a given challenge $\mathbf{c}$, the output of the lower APUF would flip by flipping challenge bit $\mathbf{c}[n/2]$ when $|\Delta_{Non-Flipping}| < |\Delta_{Flipping}|$. It is obvious that the smaller $|\Delta_{Non-Flipping}|$ (let's say $< e$), the higher the chance the output will be flipped. Since $\Delta_{Non-Flipping} \sim \mathcal{N}(0, m\sigma_b^2)$,

$$\Pr(|\Delta_{Non-Flipping}| < e) = \Phi(\frac{e}{\sqrt{m}\sigma_b}) - \Phi(\frac{-e}{\sqrt{m}\sigma_b}) = 2\Phi(\frac{e}{\sqrt{m}\sigma_b}) - 1.$$

If $m$ increases, then $\Pr(|\Delta_{Non-Flipping}| < e)$ decreases for any given small positive number $e$. This implies that the leakage of information is reduced when increasing $m$. This explains why the number of convergences to the upper APUF for case $m = 8$ is smaller than when $m = 12$. Of course, when $m$ is large enough, the attack does not work. In our simulation, we just need $m$ to be larger than 16 to prevent the attack. In our simulation, we noticed that when $m = 4$, the attack does not work. The weights of lower APUF when $m = 4$ is described in Fig. 9. We give the following possible reasoning. In the non-flipping part, there are two significant large weights and the second largest weight among the two is much smaller ($< 13$) than the largest one ($\approx 23$). Moreover, both of them are much bigger than the remaining weights. The smallest value of $|\Delta_{Non-Flipping}|$ is 10 and largest value is 36. This implies that $|\Delta_{Non-Flipping}|$ is always larger than $|\Delta_{Flipping}|$. It means that the leakage of the output of the upper APUF is very small and thus, the attack does not work. Actually in this case, the SAC property for $m = 4$ is 0.39 which is smaller than ideal SAC of 0.5.

***Classical Machine Learning Attacks on XOR APUFs and IPUFs*** In this subsection we conduct experiments to verify the claim that the the model complexity of an $(x, y)$-IPUF is similar to that of a (y+$\frac{x}{2}$)-XOR APUF. In this

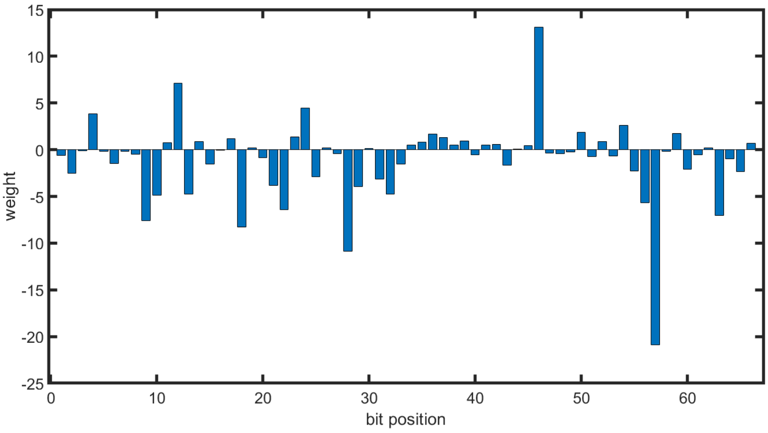


Fig. 9: Weights distribution of a lower APUF in one (1,1)-IPUF where in the second half of the lower APUF the number of large weights, whose absolute value is greater than 3, is constrained to 4.

experiment we generate component APUFs using the random placement design strategy to construct IPUFs and XOR APUFs.

We measured 200,000 CRPs for each XOR APUF and IPUF. We then used CMA-ES to optimize each model given the 200,000 CRPs. To further reduce the influence of noisy CRPs in this attack, for each CRP we did a majority voting from 11 repeated measurements. We ran CMA-ES on this training dataset 10 times to avoid the possible failure introduced by the probabilistic nature of the algorithm. The results are shown in Fig. 10

***SAC Property of (3,3) IPUF*** We tested SAC property of an implemented (3,3)-IPUF, where each component APUF is generated by random placement. The shape is shown in Figure 11, which is similar to Figure 6.

***Reliability of the IPUF with respect to Interposed Position*** We selected 22 and 23 unique component APUFs to construct a (20,2)-IPUF and a (20,3)-IPUF respectively on the FPGA. We evaluated how the noise rate was affected by changing the interposed positions in the lower XOR APUFs. We tested 5 different interposed bit positions (0, 16, 32, 48, 64). The noise rate of the (20,2)-IPUF and the (20,3)-IPUF with respect to interposed position are shown in Fig. 12. This follows the same trend as the simulation result in Fig. 8. Note that the reliability is equal to (1-noise rate) or $(100\% - noiserate)$ in percentage.

***Reliability of IPUF under Temperature Variation*** We used a (3,3)-IPUF for reliability testing under temperature variation, where each APUF is created by random placement, and have good uniformity and uniqueness. We measured 1,000 CRPs from this IPUF under 25 degree Celsius as the reference CRPs.

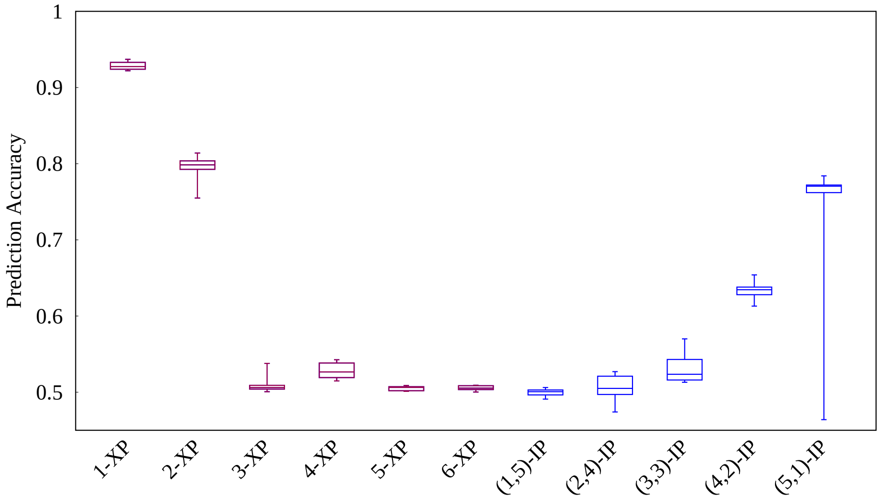


Fig. 10: Results of classical CMA-ES attacks on XOR APUFs and IPUFs.

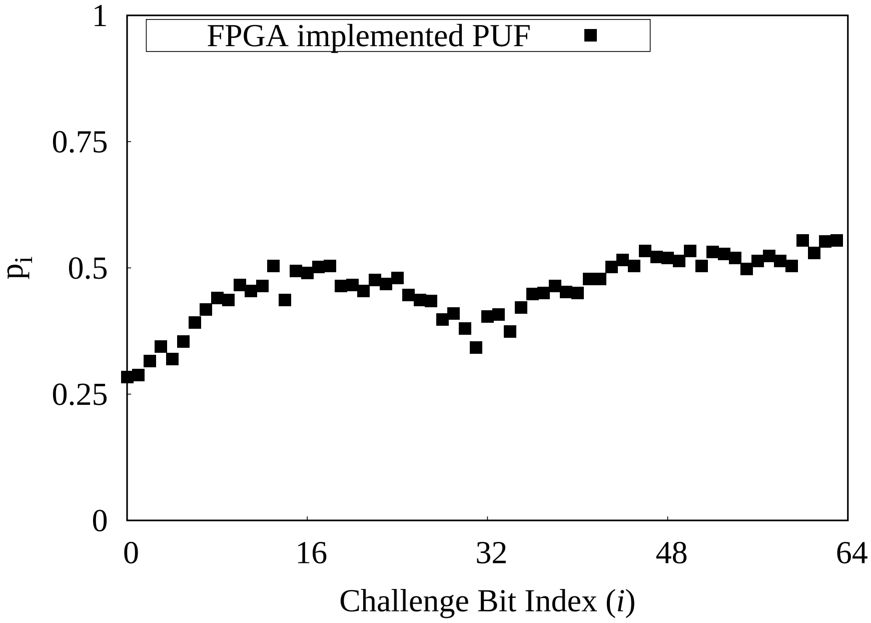


Fig. 11: SAC property of 64-bit (3,3)-IPUF with real data.

We then measured the same 1,000 CRPs again under 70 degree Celsius. The

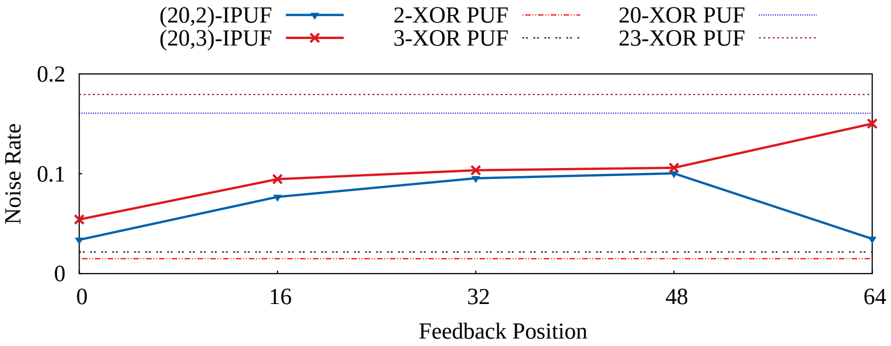


Fig. 12: Reliability with respect to different feedback position.

error rate introduced by 70 degrees and 0 degrees Celsius is 2.1% and 1.4%, respectively.

## 8 Related and Future Works

In [17], the authors introduced the Feed-Forward APUF (FFA). The security of the FFA was analysed in [26]. In essence, the FFA is conceptually similar to the IPUF but it has one or more multiple interposed bits. One way to consider the FFA design is if an APUF is divided into multiple sub-APUFs and then theirs outputs are used as interposed bits. The difference between the FFA and the (1,1)-IPUF is the construction of interposed bits. From the structure of the FFA, the reliability of the FFA depends on the reliability of all the interposed bits. When the size of the sub-APUFs are small, its reliability is not high. Hence, the reliability of the FFA is poor compared to an APUF or (1,1)-IPUF. Moreover, we need to compute the value of all the interposed bits of the FFA before we can produce the final output of FFA. The implementation of an FFA is more complicated when compared to an APUF or (1,1)-IPUF. For these reasons we consider the (1,1)-IPUF to develop the $(x, y)$-IPUF. However, it may be of interest to study the security of the $x$-XOR FFA structure in context of reliability based machine learning attacks. We suggest this analysis as a possible future work.

In [34], the authors suggest increasing the reliability of an $x$-XOR APUF to deter reliability based machine learning attacks. They accomplish this by using majority voting on the output of each APUF component in an $x$-XOR APUF. However, there are three key design problems with this approach. First, in this approach the latency of the output scales with the number of majority votes used for each APUF [34] due to majority voting. Second, majority voting requires a larger trusted computing base in the form of volatile memory and digital circuitry to store the votes. Third, this design only makes the CMA-ES reliability attack more difficult to execute, it does not completely stop it. For example, changing the temperature could make the XOR APUF less reliable and easier to attack, even when majority voting is used. Compared to the majority voting XOR APUF, our $(x, y)$-IPUF is theoretically and experimentally proven to completely stop the CMA-ES reliability attack when the interposed bit position is properly chosen. Our IPUF does not require iterative evaluation of the PUF circuitry like majority voting does, resulting in an order of magnitude higher throughput. Lastly, our IPUF design does not require majority voting circuitry, giving it a smaller hardware footprint and a smaller trusted computing base.

Currently, there is already one open source PUF simulation and machine learning attack library called Pypuf on GitHub. Pypuf contains Logistic Regression and PAC learning attacks on a variety of PUFs including APUFs, XOR APUFs, Lightweight Secure PUFs and majority voting XOR APUFs. Our PUF library contains Logistic Regression and CMA-ES attacks on APUFs, XOR APUFs, and IPUFs. Our library also provides hardware implementation of XOR APUFs and IPUFs.

## 9 Conclusion

In this paper, we comprehensively analyzed the CMA-ES reliability attack to enhance it and to create a more secure PUF design. Based on our findings we developed three main contributions. First, we improved the original CMA-ES reliability attack by changing the way the reliability information was computed. Using this enhancement, we are able to model both the APUF and XOR APUF with greater accuracy than the original attack.

Our second contribution is a new PUF design that is secure against the CMA-ES reliability attack, the $(x, y)$-IPUF. Through both theory and experiments, we showed that the $(x, y)$-IPUF is secure against the CMA-ES reliability attack, classical machine learning attacks and attacks that approximate our design as an XOR APUF. We also analytically and experimentally verified our claim that the $(x, y)$-IPUF is twice as reliable as an $(x + y)$-XOR APUF. In comparison to an XOR APUF, in this paper we proved that the IPUF is not vulnerable to the strongest known reliability based machine learning attack (white box derivative free) and the strongest known classical machine learning attack (white box derivative based). Since the IPUF has more advantages in terms of security, reliability and hardware overhead compared to the XOR APUF, **it implies that the IPUF can be considered a standard design or primitive replacement for the XOR APUF**.

Our final contribution is publicly available source code for all our IPUF, XOR APUF and APUF attack simulations written in Matlab and C#. We also provide source code for the FPGA implementation of the XOR APUF and IPUF, together with scripts to generate the constraint file for placing look up tables on an FPGA. Code in assembly for the FPGA processor to interface with a PUF is also given. All codes for this paper can be found on Github: Defense Attack (DA) PUF Library.

## Acknowledgment

This project was supported in part by the AFOSR MURI under award number FA9550-14-1-0351, and in part funded by an NSF grant CNS-1617774 "Self-Recovering Certificate Authorities using Backward and Forward Secure Key Management".

## References

1. A Fourier Analysis Based Attack against Physically Unclonable Functions (2018)
2. Becker, G.T.: On the Pitfalls of using Arbiter-PUFs as Building Blocks. IACR Cryptology ePrint Archive 2014, 532 (2014)
3. 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)
4. 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)
5. 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)
6. 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)
7. 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)
8. Digilent: Nexys 4 DDR Reference Manual (Apr 2016), https://goo.gl/g8A4e8, [Accessed Feb., 2018]
9. 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)
10. 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)
11. Ganji, F., Tajik, S., Seifert, J.P.: PAC learning of arbiter PUFs. Journal of Cryptographic Engineering 6(3), 249–258 (2016)
12. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM CCS (2002)
13. Hori, Y., Yoshida, T., Katashita, T., Satoh, A.: Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In: Proceedings of International Conference on Reconfigurable Computing and FPGAs (ReConFig). pp. 298 –303 (dec 2010)
14. Jin, C., Herder, C., Ren, L., Nguyen, P.H., Fuller, B., Devadas, S., van Dijk, M.: FPGA Implementation of a Cryptographically-Secure PUF Based on Learning Parity with Noise. Cryptography 1(3), 23 (2017)
15. 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)

16. 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)
17. Lim, D.: Extracting Secret Keys from Integrated Circuits. Master's thesis, MIT, USA (2004)
18. 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)
19. Machida, T., Yamamoto, D., Iwamoto, M., Sakiyama, K.: A New Mode of Operation for Arbiter PUF to Improve Uniqueness on FPGA. In: Proc. of Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 871–878 (2014)
20. 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)
21. Maiti, A., Gunreddy, V., Schaumont, P.: A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. IACR Cryptology ePrint Archive 2011, 657 (2011)
22. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing Techniques for Hardware Security. In: Proc. of IEEE International Test Conference(ITC). pp. 1–10 (Oct 2008)
23. 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)
24. 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)
25. Pappu, R.S.: Physical one-way functions. Ph.D. thesis, Massachusetts Institute of Technology (March 2001)
26. 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)
27. 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)
28. 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)
29. Sölter, J.: Cryptanalysis of Electrical PUFs via Machine Learning Algorithms. Master's thesis, Technische Universität München (2009)
30. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC. pp. 9–14 (2007)
31. 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)

32. 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)
33. 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
34. Wisiol, N., Graebnitz, C., Margraf, M., Oswald, M., Soroceanu, T., Zengin, B.: Why Attackers Lose: Design and Security Analysis of Arbitrarily Large XOR Arbiter PUFs
35. 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)