Modeling Attack Resilient Reconfigurable Latent Obfuscation Technique for PUF based Lightweight Authentication

Yansong Gao, Student Member, IEEE, Said F. Al-Sarawi, Member, IEEE, Derek Abbott, Fellow, IEEE
Ahmad-Reza Sadeghi, Member, IEEE, and Damith C. Ranasinghe, Member, IEEE

Abstract—Physical unclonable functions (PUFs), as hardware security primitives, exploit manufacturing randomness to extract hardware instance-specific secrets. One of most popular structures is time-delay based Arbiter PUF attributing to large number of challenge response pairs (CRPs) yielded and its compact realization. However, modeling building attacks threaten most variants of APUFs that are usually employed for strong PUF-oriented application—lightweight authentication—without reliance on the securely stored digital secrets based standard cryptographic protocols. In this paper, we investigate a reconfigurable latent obfuscation technique endowed PUF construction, coined as OB-PUF, to maintain the security of elementary PUF CRPs enabled authentication where a CRP is never used more than once. The obfuscation—determined by said random patterns—conceals and distorts the relationship between challenge-response pairs capable of thwarting a model building adversary needing to know the exact relationship between challenges and responses. A bit further, the obfuscation is hidden and reconfigured on demand, in other words, the patterns are not only invisible but also act as one-time pads that are only employed once per authentication around and then discarded. As a consequence, the OB-PUF demonstrates significant resistance to the recent revealed powerful Evaluation Strategy (ES) based modeling attacks where the direct relationship between challenge and response is even not a must. The OB-PUF’s uniqueness and reliability metrics are also systematically studied followed by formal authentication capability evaluations.

Index Terms—Reconfigurable OB-PUF, obfuscation, CMA-ES attacks, hardware security, authentication.

1 INTRODUCTION

Most current security solutions resort to storing a digital secret key in non-volatile memory (NVM) that remains hidden to an adversary. However, this requirement is difficult to realize in practice. Firstly, invasive techniques, e.g., semi-invasive methods and side channel attacks, may extract valuable key information [1], [2]. Secondly, malware such as Trojan horses or viruses can read out and transfer keys unbeknownst to users [2]. Furthermore, lightweight and mobile devices such as smart cards, Internet-of-Things (IoT) devices and radio-frequency identification (RFID) labels may have no dedicated room for key protection as functionality and cost consideration mostly dominate security requirements in commercial scenarios [2].

The PUF offers a cost-effective alternative to building security services such as authentication or identification without the need to store secure keys in NVM [3], [4], [5]. A PUF extracts secrets on demand from unclonable and uncontrollable process variations resulting from manufacturing processes when the PUF is stimulated by an input signal (challenge).

Among all silicon based PUFs, Arbiter PUF (APUF) [3], [6] receives extensive attentions as it holds the promise to serve as a strong PUF that yields an exponential number of challenge response pairs (CRPs). This desirable property enables a lightweight authentication mechanism based on CRPs. Though recent modeling attacks threaten such an elementary strong PUF-oriented application [7], [8], [9], [10], the pursuit of a strong PUF-oriented lightweight entity authentication circuit has persisted and such a realization is regarded as an open problem in the face of plausible modeling attacks, [11], [10], [12], [13]. The most recent work from Yu et al. use a lockdown technique to prevent modeling attacks by enforcing and upper-bounding the available number of CRPs by an adversary in a trade-off of the available authentication rounds [12].

We continue the investigation of a such strong PUF to support a lightweight authentication protocol and aim to eliminate the need to limit authentication rounds in [12] while showing heuristic security against modeling attacks. This article builds upon our preliminary work [5] to construct an OB-PUF. The OB-PUF obscures the relationship between the challenges and responses using a selected pattern vectors—shall be clear in Section 3—securely generated by the trusted party—server—and the OB-PUF embedded device. By exploiting the proposed reconfigurable obfuscation technique, the reconfigurable OB-PUF shows significantly increased resilience to modeling attacks, even the most powerful ES attacks where knowledge of direct
relationship between challenge and response is not a must. We summarize the contributions of this work below:

- We hide and reconfigure the obfuscation—determined by the pattern vectors—between the challenge and response, this prevents recent revealed powerful ES based modeling attacks.
- We further implement response obfuscation besides the challenge obfuscation already considered in our previous work [5] in order to balance the Hamming weight of the obfuscated responses, which eliminates potential ES attacks based on Hamming weight enabled fittest objective as detailed in Section 5.3.3.
- We develop a server-aided authentication mechanism that exploits the resourcefulness of a server to not only conduct successful authentication but reconfigure the obfuscation between the challenge and response on demand. We evaluate the authentication capability of OB-PUF primitives through a systematic study of its inter-distance and intra-distance, and subsequently, its false acceptance rate and false rejection rate in the authentication mechanism. In comparison with the most recent work [12], the OB-PUF unlimits the available authentication rounds.
- We evaluate the heuristic security of reconfigurable OB-PUF primitives against modeling attacks, specifically, recently revealed powerful ES attacks.

The remainder of the paper is organized as follows: Section 2 introduces background and related work. Section 3 presents the proposed challenge-response obfuscation method and the corresponding method to recover the challenge and response and details their implementations. Section 4 systematically analyses the authentication power/capability of the OB-PUF under different cost, security and performance settings; Section 5 executes modeling attack tests to evaluate the enhanced security of reconfigurable OB-PUFs through extensive studies; and Section 6 concludes this paper.

2 BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Physical Unclonable Functions

When a PUF is stimulated by a challenge (input), C, a corresponding response (output), R, will be generated and determined by \( f(C) \), where \( f(\cdot) \) is a physical function that is unique and inherent to each device. Given the same challenge, C, different PUFs with the same design produce diverse responses, R. The challenge, C, and its corresponding response, R, are commonly referred to as a challenge response pair (CRP). Over the years, a number of PUF structures have been proposed, built and analyzed. Popular PUF designs include time delay based Arbiter PUF (APUF) [13], [14], [15], Ring-Oscillator PUF (RO-PUF) [3]; Memory-based PUFs leveraging device mismatch such as SRAM PUF [15], [17], [18]. Comprehensive reviews of different PUF architectures are referred to [19], [11], [20].

Examples of PUFs that build upon linear additive units include APUF [1], [3], [21] and k-sum (RO-sum) PUF [22], [19]. They have one key desirable feature, generating exponential number of CPRs. Further, the hardware implementation is simple and the area overhead is small, especially the

APUF. However, the major shortcoming is their vulnerability to modeling attacks. From a modeling attack perspective, both architectures have the same topology (constructed from linear additive blocks), therefore in this paper, the APUF is used for demonstration and description. Our work exploits the vulnerability of APUFs to modeling attacks. Therefore, we introduce the APUF next.

2.2 Modeling APUF

The APUF consists of \( k \) stages of two 2-input multiplexers as shown in Fig. 1 or any other units forming two signal paths. To generate a response bit, a signal is applied to the first stage input, while the challenge \( C \) determines the signal path to the next stage. The input signal will race through each multiplexer path (top and bottom paths) in parallel with each other. At the end of the APUF architecture, an arbiter, e.g., a latch, determines whether the top or bottom signal arrives first and hence results in a logic ‘0’ or ‘1’ accordingly.

It has been shown that an APUF can be modelled via a linear additive model because a response bit is generated by comparing the summation of each time delay segment in each stage (two 2-input multiplexers) depending on the challenge \( C \), where \( C \) is made up of \((c_1|c_2|...|c_k)\) [8], [6], [7]. Following the notations in [8], [7], the final delay difference \( t_{diff} \) between these two paths is expressed as:

\[
t_{diff} = \omega^T \Phi, \tag{1}
\]

where \( \omega \) and \( \Phi \) are the delay determined vector and the parity vector, respectively, of dimension \( k + 1 \) as a function of \( C \). We denote \( \sigma_i^{1/0} \) as the delay in stage \( i \) for the crossed \((c_i = 1)\) and uncrossed \((c_i = 0)\) signal path through the multiplexers, respectively. Hence \( \sigma_i^1 \) is the delay of stage \( i \) when \( c_i = 1 \), while \( \sigma_i^0 \) is the delay of stage \( i \) when \( c_i = 0 \). Then

\[
\omega = (\omega^1, \omega^2, ..., \omega^k, \omega^{k+1})^T, \tag{2}
\]

where

\[
\omega^1 = \frac{\sigma_i^0 - \sigma_i^1}{2}, \omega^j = \frac{\sigma_{i-1}^0 + \sigma_i^1 + \sigma_i^0 - \sigma_{i+1}^1}{2} \quad \text{for all } i = 2, ..., k.
\]

\[
\omega^{k+1} = \frac{\sigma_k^0 + \sigma_k^1}{2}, \text{ also }
\]

\[
\Phi(C) = (\Phi^1(C), ..., \Phi^k(C), 1)^T, \tag{3}
\]

where \( \Phi^j(C) = \Pi_{i=2}^{j-1}(1 - 2c_i) \) for \( j = 1, ..., k \). This \( \Phi(C) \) is also referred as the challenge feature.

Here we can see that the \( \omega \) encodes the delays for the subcomponents in the APUF stages, while the \( \Phi \) is a challenge feature as a function of \( c_1, ..., c_k \). The delay difference, \( t_{diff} \), is the inner product of \( \omega \) and \( \Phi \). If \( t_{diff} \) is greater than 0, the response bit is ‘1’, otherwise, the response bit is ‘0’.
2.3 CMA-ES Attacks

Recently, Becker et al. extensively investigated the CMA-ES based modeling attacks \[9, 10\]. In general, to attack an APUF, initially a randomly generated $\omega$, that is a $k + 1$ length delay time vector, imitates the time delay model of an APUF. Such a $\omega$ serves as a parent, many offspring delay vectors are generated based on the parent and some random mutations, e.g., random modifications of the parent delay vector. According to a fitness metric, the best child which maximize the fitness metric is determined and served as the parent for the next generation. This process is repeated until a delay time vector of sufficient accuracy is formed.

In general, a reasonable or good fitness metric must be defined—evaluating the goodness of the solution—to guide the ES algorithm to the best delay time vector despite the absence of a direct relationship between challenges and responses.

2.4 Related Work

There are several approaches to increase the complexity of the task faced by an adversary to perform modeling attacks. One approach is to add nonlinearity, such as XOR-APUFs \[6, 7\] or usage of Feed Forward Arbiter PUF \[6, 23\] and the Lightweight Secure PUF \[24\]. Unfortunately, increasing the complexity of model building attacks through the integration of more non-linear elements also significantly decreases the reliability of the PUF \[3, 23, 8\]. Urich et al. show that all above strong PUF structures can be successfully attacked up to a certain size and complexity when knowing the exact relationship between challenges and responses \[7, 8\].

Unlike the previous solutions, two recent proposals \[25, 26\] have demonstrated an alternative approach by hiding the direct relationship between a given challenge $C$ and its response $R$. These two proposals in fact make use of the modeling attacks to build accurate models of APUFs by the trusted authority—the server—and then destroy the direct access to query direct challenge and response pairs, where only distorted challenge and response pairs are exposed afterwards \[25, 25, 27\].

In the first proposal \[25, 27\], the PUF integrated device, coined as Slender PUF, only reveals a subset of responses generated on the device for authentication, while the server can search and match the received substring since it can emulate PUF responses with its PUF model. In detail, a subset string of a full-length PUF response is randomly selected by the Slender and is subsequently padded with a randomly generated string to ensure the same response bit length expected from the PUF. The Slender PUF then sends the post-processed response to the server that uses a specific recovery method to discover the randomly selected subset string and decides the authenticity of the Slender PUF held by the end-user. In the second proposal \[26\], a decimation technique is developed to randomly eliminate response bits at the end-user with a PUF integrated device (prover) and a corresponding response recovery method is developed at the server to establish the validity of the response bit string.

In summary, these two approaches efficiently conceal the challenge and response relationship to prevent an adversary from building an accurate model of a PUF by observing those visible CRPs. Although such an obfuscation mechanism in those two works efficiently prevent conventional modeling attacks usually requiring direct relationship between a challenge and a response \[7, 8\]. However, Becker et al. recently demonstrated that they are vulnerable to Evaluation Strategy (ES) based methods, in particular covariance matrix adaptation ES (CMA-ES) as detailed in Section 2.3.

We can also consider the controlled PUF \[23\] employing an obfuscation technique. It, however, requires relatively expensive on-chip error correction, the storage of associated helper data and subsequently hash logic for obfuscation, which are not needed in \[25, 26, 27\]. In addition, the employment of helper data actually places the controlled PUF under potential security threats from ES attacks \[10\].

A recent work from Yu et al. restricts the available CRPs to an adversary using a lockdown technique \[12\]. In this context, an adversary is faced with the difficulty of gaining enough training materials—CRPs—to learn an accurate model. This still allows lightweight authentication applications by upper-bounding available number of secure authentication rounds over the device’s lifetime. Notably, this lockdown technique also takes advantage of the resource-rich server to build underlying APUF models during the secure enrollment phase, similar to \[25, 26, 27\]. Also noting that all works \[25, 26, 27, 12\] reap benefits from exploiting on-chip nonce or random number generator (RNG).

We build upon the desirable features of previous works \[26, 25, 27, 12\] such as: i) enrolling the APUF model during the secure enrollment phase rather characterizing CRPs and storing them, especially when large number of authentication rounds are required; ii) obfuscation to remove direct relationship between CRPs; iii) device nonce. Our obfuscation method distorts relationship between a challenge and a response using a latent pattern that can be reconfigured (nonce). Such a technique results in an OB-PUF design that: i) shows heuristic security against most powerful ES attacks \[9, 10\]; ii) eschews the need to restrict number of authentication rounds in \[12\]; iii) removes reliance on on-chip error-correction logic, computing and storing associated helper data and subsequent hashing logic as in \[23\].

3 Challenge-Response Obfuscation and Recovery

Fig. 2 illustrates the overall architecture of our challenge-response obfuscated PUF—the OB-PUF. The OB-PUF is built upon the PUF block consisting of $n_{\text{ins}}$ APUFs that share the same full length challenge $C$. The challenge stimulated to the OB-PUF is a partial challenge $C_{OB}$ of $k - m$ bits and the generated obfuscated response $R_{OB}$ is of $n_{\text{ins}}$ bits. A partial
Figure 3. An example of challenge-response obfuscation. The pattern vector \((S_{C1}, S_{R1})\) is randomly chosen by the RNG. The \(S_{C1}\) vector determines the formation of the full length challenge \(C_1\), by controlling positions and values of the \(m = 3\) bits that can be inserted into a partial challenge \(C_{OB}\). The \(S_{R1}\) vector with length of \(n_{ins}\) is XOR-ed with \(R_1\). In general, it can be seen that one \(C_{OB}\) can result in one of \(p = 4\) possible full length challenges, \(C_1, ..., C_4\), and hence one of \(p\) possible corresponding responses, \(R_1, ..., R_4\). The response \(R_i\) is further obfuscated by XOR-ing with a specific \(S_{RI}\) vector to obtain the obfuscated response \(R_{OB_i}\), according to the selected pattern vector \(i\). The observable obfuscated response \(R_{OB} = R_{OB_i}\), determined by the chosen pattern vector \((S_{C1}, S_{Ri})\), is sent from the OB-PUF. Note pattern vectors are latent and reconfigurable, specifically, this work focuses on reconfiguring the inserted values in \(S_{RI}\) vector.

We first use a fixed pattern vector to make the descriptions more readable without affecting analyses on the OB-PUF except modeling attacks analyses. The elaborate way in which the pattern vector, more precisely, the inserted values investigated in this paper, are hidden and reconfigured on demand by using APUF responses produced within the pattern vector reconfiguring block will be fully described in Section 3.6.

As an example, consider an OB-PUF with four pattern vectors shown in Fig. 3 where the PUF block consists of four APUFs. A partial challenge is applied to the challenge-response control block of the OB-PUF. Assuming the RNG selects pattern vector 1, the full length challenge \(C_1\) is formed by inserting logic values of \(\{0', 1', 0\}'\) into the partial challenge at the respective bit positions of 1, 2, and 3. Subsequently, the formed full length challenge \(C_1\) from the challenge-response control block is applied to the PUF block. Recall, the PUF block comprises of four basic PUF instances implemented in parallel. These four basic PUF instances share the same full length challenge \(C_1\) in order to generate a 4-bit response simultaneously—the reason is described in Section 3.3. After the response \(R_1\) is generated from the PUF block, it is further processed by the challenge-response control block. Here \(R_1\) is XOR-ed with the string ‘0101’ in order to obtain the response \(R_{OB1}\). Then \(R_{OB1}\) is sent out from the challenge-response control block as the obfuscated response \(R_{OB}\) of the OB-PUF.

The central concept behind the challenge-response obfuscation is the hiding of the pattern vector chosen by the RNG. As a consequence, the full length challenge \(C_i\), \(i \in \{1, ..., p\}\), and the corresponding response \(R_i\) are not only hidden from the adversary but also the server, but only the server can recover it by implementing the challenge-response recovery method, which is described in detail in Section 3.5.

In the following subsections, first, we introduce relevant notations and definitions. Secondly, we introduce the challenge-response obfuscation employed to obfuscate the full length challenge applied to the PUF block inside the OB-PUF and the response from the underlying PUF block. Thirdly, we present the challenge-response recovery method to enable the server to discover the selected pattern vector by observing the obfuscated response, and hence, the obfuscated full length challenge and its corresponding response.

### 3.1 Preliminaries

Before delving into the details and evaluation of the OB-PUF, we give a number of useful definitions and then introduce several notations where the parameters used in the following descriptions are listed in Table 1.

As the Hamming distance (HD) and fractional Hamming distance (FHD) are often referred to in this work, we define them below, followed by definitions of the mean of pairwise HD and FHD.

**Definition 1. Hamming distance.** For binary strings \(X_1\) and \(X_2\) with the same length \(l\), the HD between them is defined as:

\[
    f_{HD}(X_1, X_2) = \sum_{i=1}^{l} X_1 \oplus X_2.
\]

**Definition 2. Fractional Hamming distance.** Built upon (4), the fractional Hamming distance (FHD) is defined as:

\[
    f_{FHD}(X_1, X_2) = \frac{f_{HD}(X_1, X_2)}{l}.
\]

**Definition 3. Mean of pairwise HD/FHD.** Given a collection \(X\) that consists of many binary strings, the mean of pairwise HD of \(X\) is defined as:

\[
    f_{meanHD}(X) = \text{mean}\{f_{HD}(X_i, X_j)\},
\]

where these two binary strings \(X_i \in X\), \(X_j \in X\) and \(i \neq j\).
And the mean of pairwise FHD is defined as:

\[ f_{\text{meanFHD}}(\mathbf{x}) = \frac{f_{\text{meanHD}}(\mathbf{x})}{l} \quad (7) \]

Now consider the OB-PUF illustrated in Fig. 2 and 3 formally, we can define a pattern vector as a vector chosen from a set, \( \{ (\mathbf{S}_C; \mathbf{S}_R) \}, i \in \{1, ..., p \} \). Here the \( \mathbf{S}_C \in \mathbf{S}_C \), where \( \mathbf{S}_C = \{ \mathbf{S}_{C1}, ..., \mathbf{S}_{Ci} \} \) determines the positions and values inserted into a partial challenge \( \mathbf{C}_{OB} \) to form a possible full length challenge \( \mathbf{C}_i \). The \( \mathbf{S}_R \in \mathbf{S}_R \), where \( \mathbf{S}_R = \{ \mathbf{S}_{R1}, ..., \mathbf{S}_{Ri} \} \) and the length of \( \mathbf{S}_{Ri} \) is \( n_{ins} \) is the string that is XOR-ed with the response \( \mathbf{R}_i \) obtained from the PUF block in order to gain the obfuscated response.

There are \( p \) pattern vectors, therefore, there are \( p \) possible full length challenges given a partial challenge. Then given a pattern vector, \( i \), the formed full length challenge \( \mathbf{C}_i \) applied to the PUF block is from a set \( \mathcal{C} \), where \( \mathcal{C} = \{ \mathbf{C}_1, ..., \mathbf{C}_p \} \). Consequently, the corresponding response \( \mathbf{R}_i \in \mathcal{R}, \mathcal{R} = [\mathbf{R}_1, ..., \mathbf{R}_p] \), is generated from the PUF block. The obfuscated response \( \mathbf{R}_{OB} \) from the OB-PUF is a function of \( \mathbf{R}_i \), where \( \mathbf{R}_{OB} = \mathbf{R}_i \oplus \mathbf{S}_{Ri} \). Therefore, the final output \( \mathbf{R}_{OB} = \mathbf{R}_{OB} \) of the OB-PUF is from a collection of \( \mathcal{R}_{OB} \), where \( \mathcal{R}_{OB} = [\mathbf{R}_{OB1}, ..., \mathbf{R}_{OBp}] \). In other words, the final output—the observable obfuscated response—is actually determined by the randomly selected \( i \)th pattern vector.

### 3.2 Challenge Obfuscation

The partial challenge intentionally eliminates \( m \) bits from a full length challenge. From a machine learning perspective, not only do we eliminate \( m \) features but also eliminate them from randomly selected positions. When such an incomplete input feature vector is employed to train a model, the missing features decrease the accuracy of the learned model. In Following, we describe the design of pattern vectors and realization of challenge obfuscation.

As for the OB-PUF, a partial challenge gives \( p \) possible full length challenges, \( \mathbf{C}_1, ..., \mathbf{C}_p \). Hence, there are \( p \) possible responses, \( \mathbf{R}_1, ..., \mathbf{R}_p \). According to our experiments, if the mean of pairwise HD of possible full length challenges \( \mathcal{C} \)—see Definition 3—is small, the mean of pairwise HD of possible responses \( \mathcal{R} \) will also be small.

As an exemplary demonstration, we have randomly generated 5000 \( \mathbf{C}_{OB} \) and form the four possible full length challenges according to one extreme case, where we inserted all values to the first three positions in each pattern vector—note that we do not insert the values to the positions that are optimal positions as shown in Fig. 3. We got a 3.1% mean of pairwise FHD of four possible full length challenges, consequently, the mean of pairwise FHD of four possible responses is only 1.1%. In essence, recall 3, this extreme position selection remains \( \Phi^4(\mathcal{C}), ..., \Phi^4(\mathcal{C}), 1 \) for all four possible full challenge fractures to be same. This creates a problem. Assume that the OB-PUF does not further implement response obfuscation and the mean of pairwise HD of \( \mathcal{C} \) is very small. Given one possible response \( \mathbf{R}_i \in \mathcal{R} \), for a \( \mathbf{C}_{OB} \), is directly exposed as the response of OB-PUF. Now it does not matter which pattern vector is chosen to construct a full length challenge, an adversary can select any possible full length challenge from \( \mathcal{C} \) and the exposed \( \mathbf{R}_i \) to train a model. This is because, by virtue of the small HD between full length challenges in \( \mathcal{C} \), the mean of pairwise HD of possible responses \( \mathcal{R} \) is also small. In other words, it is reasonable for an adversary to select any \( C_j \in \mathcal{C}, j \in \{1, ..., p\} \), to pair with the observable \( \mathbf{R}_i \) as a useful CRP to train a model.

To mitigate the above security concern, the mean of pairwise HD of possible responses \( \mathcal{R} \) for a given partial challenge should be maximized. Alternatively, the mean of pairwise HD of possible full length challenges \( \mathcal{C} \) should be maximized.

Then, it follows that to maximize the mean of pairwise HD of possible responses \( \mathcal{R} \), the pattern vector used to generate full length challenges should be carefully designed. The four pattern vectors, \( p = 4 \) as an example, listed in Fig. 3 is one strategy to maximize the mean of pairwise HD of possible full length challenges \( \mathcal{C} \).

A generalized pattern vector design for challenge obfuscation is illustrated in Fig. 5. In Fig. 4, it shows the distribution of the mean of pairwise FHD of possible full length challenges \( \mathcal{C} \) and the distribution of the mean of pairwise FHD of corresponding responses \( \mathcal{R} \). To generate the results in Fig. 4, we calculate the mean of pairwise FHD of possible full length challenge \( \mathcal{C} \) and the corresponding possible responses \( \mathcal{R} \) for each randomly generated partial challenge. The number of randomly generated partial challenges is 1,000,000, consequently, the distributions of the mean of pairwise FHD of possible full length challenges \( \mathcal{C} \) and the corresponding possible responses \( \mathcal{R} \) for 1,000,000 partial challenges are acquired, where \( k = 64 \) and \( n_{ins} = 64 \). It can be observed from Fig. 4 that the mean of pairwise FHD of possible full length challenges \( \mathcal{C} \) and the corresponding possible responses \( \mathcal{R} \) are always in vicinity of 50%, and, hence, indicating that the mean of pairwise FHD of \( \mathcal{C} \) and \( \mathcal{R} \) are maximized. Notably, a large mean pairwise FHD of \( \mathcal{R} \)—close to 50%—helps increase the ability to discover the hidden full length challenge by a server or a verifier armed with a model of OB-PUF as will be discussed in Section 3.3.

### 3.3 Response Obfuscation

Response obfuscation further increases the mean pairwise HD of possible responses from an OB-PUF. Most importantly, it is employed to prevent an adversary with full access to the OB-PUF possibly discovering and employing those partial challenges, where the responses generated from an APUF given a partial challenge is only determined
The reason lies in the fact that the selected pattern vector always generates the same response, an adversary can randomly select pattern vectors. In summary:

1) The challenge obfuscation pattern vector $S_C$ must be designed to maximize the pairwise HD of possible full length challenges $C_1, ..., C_p$ for a given $C_{OB}$.

2) The response obfuscation pattern vector $S_R$ must be designed to maximize the mean of pairwise HD of $S_R$ to be 50% and to ensure the percentage of ‘1’ in a given bit position of $S_R$ corresponding to every pattern vector to be approximately 50%.

### 3.5 Challenge-Response Recovery

When a server receives an obfuscated response, $R_{OB}$, from an OB-PUF, the server can emulate all $p$ possible obfuscated responses $R_{OB} = [R_{OB_1}, ..., R_{OB_p}]$, for a given $C_{OB}$, according to the pattern vectors known by the server. As illustrated in Fig. 6, the server compares each emulated obfuscated response $R'_{OB_i}, i \in \{1, ..., p\}$ with the received $R_{OB}$ from the user and accept if there exists one emulated obfuscated response $R'_{OB_i}$ that matches the received $R_{OB}$. The matching criterion is detailed in Section 4.1 along with the OB-PUF based authentication protocol.

### 4 Authentication Application and Analysis

In this section, first, we summarize a parameter-based authentication protocol applicable to the OB-PUF primitive. In addition, the authentication protocol can further harness a server-aided computation method to improve the OB-PUF’s performance. Next, we introduce the general formula for establishing the authentication capability of PUFs and evaluate the crucial parameters, inter-distance and intra-distance, necessary to assess the authentication capability of our OB-PUFs. Then we quantitatively analyze the OB-PUF’s authentication capability. At the end, we detail how to reconfigure the pattern vectors in authentication applications.

#### 4.1 Server-Aided Parameter Based Authentication Protocol

A variant of parameter-based authentication protocol [26] is described in detail below:

1) A server gains a specific number of CRPs of each APUF in a PUF block to train a model and securely stores the parameter model associated with a given instance of an OB-PUF. This is usually referred to as the enrollment phase. Then the direct extraction of CRPs of APUF instances are destroyed, e.g., via fusing the extraction wires.

2) Whenever the prover requests an authentication from the server. The server randomly issues a $C_{OB}$ and sends it to the prover. The prover
applies the $C_{OB}$ to the OB-PUF and obtains an $R_{OB}$.

3: After receiving $R_{OB}$ from the prover, the server carries out the authentication by implementing the challenge-response recovery method. If one of the emulated obfuscated responses $R_{OB}'$ for a given $C_{OB}$ matches the received obfuscated response $R_{OB}$—see Section 3.5—the authentication is successful, and the authenticity of the physical entity is established; otherwise, if none of the emulated obfuscated responses $R_{OB}'$ matches to the received $R_{OB}$, the authentication fails. A specific $C_{OB}$ is used no more than once to prevent replay attacks.

Notably, this parameter based authentication can further exploit resourcefulness of the server by implementing the following server-aided method.

Determining a Match. In the example in Fig. 6 a match is determined if the criterion $f_{HD}(R_{OB}', R_{OB}) = 0$ is satisfied when performing the step 3 of the authentication protocol. This occurs when the pattern vector that the server selects for emulation is the same as the pattern vector randomly chosen in the physical OB-PUF. Otherwise, if none of the emulated $R_{OB}'$ matches the received $R_{OB}$. This authentication around is rejected. In general, a match criterion does not always require $f_{HD}(R_{OB}', R_{OB}) = 0$. As long as $f_{HD}(R_{OB}', R_{OB})$ is lower than a mismatch tolerance $n_{mismatch}$ defined below, the server can accept the authenticity of the received obfuscated response.

Definition 4. Mismatch tolerance $n_{mismatch}$. The $n_{mismatch}$ is the Hamming distance—number of mismatching bits—between an emulated obfuscated response $R_{OB}'$, and the received obfuscated response $R_{OB}$ tolerated by a match criterion. A server accepts the received obfuscated response $R_{OB}$ if $f_{HD}(R_{OB}', R_{OB}) \leq n_{mismatch}$.

In other words, $n_{mismatch}$ describes tolerated number of flipped bits between the $R_{OB}'$, and $R_{OB}$ when the server compares them. The $R_{OB}$ is accepted when the HD between one of $p$ possible emulated $R_{OB}'$ and the received $R_{OB}$ is no more than $n_{mismatch}$. We will show in Section 4.4 that relaxing match criterion by selecting a large mismatch tolerance in fact expedites the authentication process. Clearly a single $C_{OB}$ and $R_{OB}$ pair, short of an OB-CRP, is not adequate to authenticate a large population of OB-PUFs. One solution is to implement multiple OB-PUF instances sharing the same $C_{OB}$ on a device, but this results into higher area and power overhead. The alternative approach of using multiple number of challenge response pairs is more practical, and is commonly employed by PUF based authentication protocols [21]. In our authentication protocol, this implies that steps 2 and 3 are repeated multiple times. The goal with either approach is to generate adequate number of response bits to authenticate or identify a large number of OB-PUFs.

4.2 Authentication Capability

Considering our need for multiple rounds or multiple OB-CPRs, the question that we need to address remains the number of OB-CPRs required to authenticate or identify an OB-PUF from a very large population of OB-PUFs.

![Figure 7. Distributions of intra-distance and inter-distance for 32-bit response.](image)

Starting with a conventional PUF, we can establish the number of CRPs needed to authenticate a PUF by considering the inter-distance and intra-distance used to describe the distribution statistics of PUF responses. We formally define the inter-distance and intra-distance following Roel’s work [11].

Definition 5. Inter-distance. The inter-distance is a random variable describing the distance between two PUF responses from different PUF instances subjected to the same challenge, hence,

$$D_{inter} = \text{dist}(R_A; R_B)$$

where $R_A$ and $R_B$ are two responses from two randomly selected but distinct PUFs stimulated by the same challenge. The inter-distance measures the uniqueness of PUFs [11].

Definition 6. Intra-distance. The intra-distance is a random variable describing the distance between two PUF responses from the same PUF instance and subjected the same challenge, hence,

$$D_{intra} = \text{dist}(R; R')$$

where $R$ and $R'$ are two distinct responses from a randomly chosen PUF instance produced using the same challenge re-evaluated at two distinct times. The intra-distance describes a PUF’s response reproducibility.

The $\text{dist}(\cdot \cdot \cdot)$ function above can be any well-defined and appropriate distance metric over the responses. In this paper, responses are always bit vectors and the used distance metric is Hamming distance or fractional Hamming distance—see Section 3.2. In Fig. 7, we illustrate the distribution of the inter-distance and intra-distance of 32-bit length PUF responses. Considering that the evaluation of each CRP or each response bit is independent [6], both inter- and intra-distance distributions are assumed to follow a binomial distribution, $B(n, \hat{p})$, where the binomial probability estimator of inter- and intra-distance distributions are $\hat{p}_{inter}$ and $\hat{p}_{intra}$ respectively [6]. Then $\hat{p}_{inter}$ is the probability that $R_A \neq R_B$, and $\hat{p}_{intra}$ is the probability that $R \neq R'$ [6].

We are particularly interested in the right tail of intra-distance distribution and the left tail of inter-distance distribution, because these two tails describe two undesirable errors in an authentication application: i) false rejection rate (FRR); and ii) false acceptance rate (FAR). We can see that given an $n$-bit response vector, FAR expresses the probability of misidentification and therefore a high FAR is a security weakness in the authentication mechanism. On the other hand, FRR expresses the probability of misrejection of an authentic PUF and therefore a high FRR will deem the authentication scheme impractical. Given both inter- and
intra-distance follow binomial distributions, the FRR and FAR can be formally expressed in [21], [11]:

\[
\begin{align*}
\text{FRR} & = 1 - \sum_{i=0}^{n_{th}} \binom{n}{i} (\hat{p}_{\text{intra}})^i (1 - \hat{p}_{\text{intra}})^{(n-i)}, \\
\text{FAR} & = \sum_{i=0}^{n_{th}} \binom{n}{i} (\hat{p}_{\text{inter}})^i (1 - \hat{p}_{\text{inter}})^{(n-i)}.
\end{align*}
\]

We can easily envision that a larger \(n_{th}\) benefits FRR but increases the FAR, and vice versa. However, high values for both FRR and FAR are undesirable for authentication. Therefore, in practice, we want a balance between them. There exists a value for \(n_{th}\) for which both FRR and FAR are equal. We refer to this value of \(n_{th}\) as equal error threshold, \(n_{EER}\), and when both error rates are equal, we refer to this as equal error rate (EER) following Roel’s work in [11]. For a discrete distribution, there may not be a value for \(n_{EER}\) where FAR and FRR are exactly equal, and hence, \(n_{EER}\) and EER are defined as in [11]:

\[
\begin{align*}
n_{EER} & = \arg \min_{n_{th}} \{ \max \{ \text{FAR}(n_{th}), \text{FRR}(n_{th}) \} \}, \\
\text{EER} & = \max \{ \text{FAR}(n_{EER}), \text{FRR}(n_{EER}) \}.
\end{align*}
\]

Given a PUF class with binomial probability estimator \(\hat{p}_{\text{inter}}\) and \(\hat{p}_{\text{intra}}\), we can find the minimum number of CRPs, \(n\), for ensuring an acceptable EER—typically in practice, a minimum requirement is a value lower than 10–6. During an OB-PUF authentication, the server sends out \(n\) partial challenges \(C_{OB}\) and compares the received \(n\) obfuscated responses \(R_{OB}\). Then, to evaluate an adequate value \(n\) to achieve an acceptable EER, we need to statistically evaluate the \(\hat{p}_{\text{intra}}\) and \(\hat{p}_{\text{inter}}\) for our OB-PUF primitive.

### 4.3 The Binomial Probability Estimator \(\hat{p}_{\text{inter}}\) of OB-PUF

Consider two 1-bit responses produced by applying the same challenge to two distinct conventional PUFs that are randomly chosen from a population. In general, the \(\hat{p}_{\text{inter}}\) of conventional PUFs is the probability that these two 1-bit responses are different. This is normally very close to 50% since most PUFs can achieve a uniqueness very close to 50% when the layout is carefully performed. Similarly, the \(\hat{p}_{\text{inter}}\) of OB-PUF is the probability that two obfuscated responses are different, where these two obfuscated responses are generated by two randomly chosen OB-PUFs from a population given the same applied partial challenge.

Formally, we can derive a generalized expression for \(\hat{p}_{\text{inter}}\) of OB-PUFs. Consider that OB-PUF \(A\) will generate a possible \(R_{OB_A} \in [R_{OB_{A1}}, R_{OB_{A2}}, ..., R_{OB_{Ap}}]\), where \(p\) is the number of pattern vectors. Similarly, the OB-PUF \(B\) will generate a possible \(R_{OB_B} \in [R_{OB_{B1}}, R_{OB_{B2}}, ..., R_{OB_{Bp}}]\). The \(\hat{p}_{\text{inter}}\) of OB-PUFs is actually the probability \(R_{OB_A} \neq R_{OB_B}\) given by:

\[
P(R_{OB_A} \neq R_{OB_B}) = P((R_{OB_{A1}} \neq R_{OB_{B1}}) \land (R_{OB_{A2}} \neq R_{OB_{B2}}), ..., \land (R_{OB_{Ap}} \neq R_{OB_{Bp}}))
\]

\[
= P(R_{OB_A} \neq R_{OB_B}) P(R_{OB_{A1}} \neq R_{OB_{B1}}) P(R_{OB_{A2}} \neq R_{OB_{B2}}) ...
\]

\[
= P(R_{OB_A} \neq R_{OB_B}) P(R_{OB_{A2}} \neq R_{OB_{B2}}) P(R_{OB_{A3}} \neq R_{OB_{B3}}) ...
\]

\[
= P(R_{OB_A} \neq R_{OB_B}),
\]

where

\[
P(R_{OB_{Ai}} \neq R_{OB_{Bi}}) = P(R_{OB_{Ai}} \neq R_{OB_{Bi}}) \land (R_{OB_{Ai}} \neq R_{OB_{Bi}}), \ldots, \land (R_{OB_{Ai}} \neq R_{OB_{Bi}})) = P(R_{OB_{Ai}} \neq R_{OB_{Bi}}) P(R_{OB_{Ai}} \neq R_{OB_{Bi}})
\]

\[
= P(R_{OB_{Ai}} \neq R_{OB_{Bi}}), i \in \{1, 2, ..., p\}.
\]

As for OB-PUFs, the obfuscated response consists of multiple bits, therefore, \(R_{OB_{Ai}} \neq R_{OB_{Bi}}\) actually implies that the \(f_{HD}(R_{OB_{Ai}}, R_{OB_{Bi}}) > n_{\text{mismatch}}\). Consequently,

\[
P(R_{OB_{Ai}} \neq R_{OB_{Bi}}) = 1 - P(f_{HD}(R_{OB_{Ai}}, R_{OB_{Bi}}) \leq n_{\text{mismatch}})
\]

\[
= 1 - \sum_{0}^{n_{\text{mismatch}}} \frac{1}{2^n_{\text{ins}}}.
\]

Therefore,

\[
P(R_{OB_{Ai}} \neq R_{OB_{Bi}})
\]

\[
= (1 - \sum_{0}^{n_{\text{mismatch}}} \frac{1}{2^n_{\text{ins}}})^p, i \in \{1, 2, ..., p\},
\]

and,

\[
P(R_{OB_{A}} \neq R_{OB_{B}}) = (1 - \sum_{0}^{n_{\text{mismatch}}} \frac{1}{2^n_{\text{ins}}})^p
\]

\[
= (1 - \sum_{0}^{n_{\text{mismatch}}} \frac{1}{2^n_{\text{ins}}})^p.
\]

As \(\hat{p}_{\text{inter}} = P(R_{OB_{A}} \neq R_{OB_{B}})\), then \(\hat{p}_{\text{inter}}\) of OB-PUFs can be formally expressed as:

\[
\hat{p}_{\text{inter}} = (1 - \sum_{0}^{n_{\text{mismatch}}} \frac{1}{2^n_{\text{ins}}})^p.
\]

Based on [20], we can see that smaller number of PUF instances \(n_{\text{ins}}\) in a PUF block decreases \(\hat{p}_{\text{inter}}\). For example, if \(n_{\text{ins}}\) is one, then the \(\hat{p}_{\text{inter}}\) is only \(\frac{1}{2}\) given \(p = 2\). This extremely small \(\hat{p}_{\text{inter}}\) implies that the OB-PUF based authentication mechanism yields undesirably high FAR. Fortunately, inter-distance can be easily regained by increasing the number of PUF instances implemented in parallel, and simultaneously queried by the same full length challenge.

### 4.4 The Binomial Probability Estimator \(\hat{p}_{\text{intra}}\) of OB-PUF

In general, the binomial probability estimator \(\hat{p}_{\text{intra}}\) of a conventional PUF is actually the probability that two repeatedly generated 1-bit responses, given the same challenge applied to the same PUF, are different.

As for an OB-PUF, the \(\hat{p}_{\text{intra}}\) is the probability that two repeatedly generated obfuscated responses given the same full length challenge applied to the same OB-PUF are different. Considering that an OB-PUF produces an \(n_{\text{ins}}\)-bit obfuscated response, we can see that \(\hat{p}_{\text{intra}}\) is actually the probability of \(f_{HD}(R_{OB}, R_{OB}) > n_{\text{mismatch}}\).
where the $R_{OB}^1$ and $R_{OB}^2$ are two repeatedly generated obfuscated responses at two different time instances. However, it is much easier to first formulate the probability of $f_{HD}(R_{OB}^1, R_{OB}^2) \leq n_{mismatch}$ given as:

\[
\sum_{i=0}^{n_{mismatch}} \binom{n_{ins}}{i} \left(1 - \hat{p}_{intra}^{n_{ins}}\right)^i \left(\hat{p}_{intra}^{n_{ins}}\right)^{n_{ins} - i}.
\]  

\[n_{mismatch} \leq \sum_{i=0}^{n_{mismatch}} \binom{n_{ins}}{i} \left(1 - \hat{p}_{intra}^{n_{ins}}\right)^i \left(\hat{p}_{intra}^{n_{ins}}\right)^{n_{ins} - i}. \tag{21} \]

In [21], to distinguish the $\hat{p}_{intra}$ of OB-PUFs that of the underlying PUF in the PUF block, we refer to the $\hat{p}_{intra}$ of an underlying PUF as $\hat{p}_{intra}^{n_{ins}}$.

Then, formally, we can express the binomial probability estimator of intra-distance of OB-PUF, $\hat{p}_{intra}$, as:

\[
\hat{p}_{intra} = 1 - P(f_{HD}(R_{OB}^1, R_{OB}^2) \leq n_{mismatch})
\]

\[
= 1 - \sum_{i=0}^{n_{mismatch}} \binom{n_{ins}}{i} \left(1 - \hat{p}_{intra}^{n_{ins}}\right)^i \left(\hat{p}_{intra}^{n_{ins}}\right)^{n_{ins} - i}. \tag{22} \]

### 4.5 Evaluating OB-PUF Authentication Capability

We can quantitatively analyze the FAR and FRR of OB-PUF when using $n$ OB-CRPs for authentication. According to [11] and [10], the FAR and FRR are determined by binomial probability estimators $\hat{p}_{inter}$ and $\hat{p}_{intra}$, chosen $n_{th}$ and the number of OB-CRPs $n$. Notably, the binomial probability estimators of OB-PUFs, $\hat{p}_{intra}$ and $\hat{p}_{inter}$, are functions of OB-PUF implementation parameters $n_{ins}$, $p$, and $n_{mismatch}$. Hence, various combinations of these implementation parameters naturally lead to a family of OB-PUFs with differing physical features and security features such as authentication capability. Here, it is convenient to express the resulting family of OB-PUFs as OB-PUF($n_{ins}$, $p$, $n_{mismatch}$).

In Table 2, we give a quantitative evaluation of authentication power under different configurations of OB-PUF($n_{ins}$, $p$, $n_{mismatch}$). Here, $\hat{p}_{intra}^{n_{ins}}$ is selected as 5% since that is the worst-case $\hat{p}_{intra}$ reported for APUF exposed to 45°C variation of temperature and 2% supply voltage variation according to the experimental data in [21].

#### 4.5.1 Size of the PUF Block

We can see from Table 2 that the required number of OB-CRPs, $n$, to achieve a specific EER, e.g., $10^{-6}$, decreases as $n_{ins}$ increases. A larger number of PUF instances $n_{ins}$ requires a smaller $n$, which in turn requires less time for sequentially sending and receiving those OB-CRPs and thus enabling faster authentication. However, this approach will result in higher overhead costs as the silicon area needed to implement an OB-PUF increases. In practical applications where authentication period is not of strict concern, selecting a small value for $n_{ins}$ provides a lightweight primitive.

#### 4.5.2 Number of Pattern Vectors

Based on the OB-PUF(4, 2, 0) and OB-PUF(4, 4, 0), we can see that as the number of pattern vectors $p$ increases, the number of needed OB-CRPs $n_{th}$ increases. This is because the larger $p$ decreases the inter-distance according to [20], but from a security perspective, a larger $p$ achieves a higher security level, because it reduces the probability of an adversary correctly guessing the selected pattern vector, see Section 5.

#### 4.5.3 Mismatch Tolerance

Considering OB-PUF(16, 4, 0) and OB-PUF(16, 4, 1), we can see that $n$ can be smaller by using a slightly larger $n_{mismatch}$, e.g., $n_{mismatch} = 1$, recall that $n_{mismatch}$ is the number of tolerated mismatching bits when comparing an emulated obfuscated response with the received obfuscated response—see Determining a Match in Section 4. In detail, setting $n_{mismatch} = 1$ increases the difference between $\hat{p}_{inter}$ and $\hat{p}_{intra}$ when $n_{ins}$ is large, for example, more than ten as illustrated in Fig. 3 and in turn expedites the authentication process. In other words, the authentication period can be made shorter. Notably, for $n_{ins} = 16$ for both OB-PUF(16, 4, 0) and OB-PUF(16, 4, 1), the authentication period benefits from setting $n_{mismatch} = 1$. This improvement can also be understood from Fig. 7. Consider a fixed $n$ and fixed $n_{th}$, when the difference between $\hat{p}_{inter}$ and $\hat{p}_{intra}$ is larger—in other words, difference between intra-distance and inter-distance is increased, FAR and FRR will eventually decrease.

Overall, OB-PUFs can be flexibly configured to suit different application scenarios according to their performance requirements such as time to complete authentication, level of security and hardware cost constraints. If the authentication time is a concern, then using a large $n_{ins}$ and setting $n_{mismatch} = 1$ or even larger is possible. Otherwise, if the OB-PUF integrated device is resource constrained and the authentication time is not a strict concern, then using a small $n_{ins}$, e.g., two, and setting $n_{mismatch} = 0$ is possible.

#### 4.6 Reconfiguring Latent Pattern Vectors

Now we introduce how we reconfigure the pattern vectors during an authentication phase. We eschew reliance on NVMs to store reconfigured pattern vectors by generating them on demand from the underlying APUFs within the pattern vector reconfiguring block, as shown in Fig. 2.
Hence, the pattern vectors are only known by the server and therefore hidden from other parties.

This paper focuses on reconfiguring the inserted values of the pattern vectors, as shown in Fig. 3, used to form the full length challenge. It is reminded that all other information of the pattern vector such as the inserted positions and the strings used to obfuscate the responses can also be reconfigured in a similar manner. As depicted in Fig. 3 before each authentication session consisting of $n$ OB-CPRs, the inserted values with bit-length of $p \times m$ are first generated from the underlying APUFs within the pattern vector reconfiguring block. The pattern vector reconfiguring block implements logic to assess all sequentially applied $p \times m$ challenges are different. If any two challenges out of $p \times m$ challenges are different, then those bits are stored in internal registers to be used for the current session. Otherwise, random numbers are filled into the registers. We consider that the APUFs in charge of reconfiguring pattern vectors share the same $C_{OB}$ with the APUFs underlying the PUF blocks to simplify the logic implementation in practice, the difference is that the former APUFs only have $k - m$ stages. In other words, the $C_{OB}$ is the full length challenge of the APUFs underlying the reconfiguring pattern vectors block.

4.6.1 Preselection of Partial Challenges for Pattern Vector Generation.

In general, the naturally noisy feature of PUF responses prevents the stable generation of the inserted values and on-chip error correction must be applied. However, when a basic APUF is modeled/learned through a method such as support vector machines (SVM) [29], the predicted time delay $t_{diff}$ in Eq. 7—standing for the difference in delay time between the top path and the bottom path of an APUF—contains two useful information: i) $\text{sgn}(t_{diff})$ that determines the binary response; ii) the reliability of this response. If the $t_{diff}$ is always far from zero, then a response from such a challenge can be reproduced with full confidence. Recent work of Xu et al. [29] shows that judiciously determined challenges based on a learned statistical APUF model can generate error free responses even across a wide range of operating conditions as well as aging effects.

Therefore, the server is able to easily determine $p \times m$ partial challenges $C_{OB}$—noting that the $C_{OB}$ is the full length challenge of the APUFs with $k - m$ stages underlying the pattern vector reconfiguring block—that always produce error-free responses on demand. We adopt such an error free response selection method to refresh the inserted values for reconfiguring all pattern vectors at the beginning of each authentication session. It is also noticed that such a reliable response selection method can further be employed to preselect partial challenges that are used for the following $n$ OB-CRPs to enable an improved reliability of the OB-CRPs.

5 Security Evaluations

5.1 Adversary Model

We employ a common adversary model described in [8], [2], [25] when evaluating OB-PUFs’ security. In summary, the adversary is allowed to eavesdrop on the communication channel and arbitrarily apply challenges using the publicly accessible OB-PUF interface to collect an arbitrary number of CRPs. The adversary’s goal is to learn an accurate model of the OB-PUF using modeling attacks.

We focus on recently revealed CMA-ES based modeling attacks where a direct relationship between a challenge and a response is not required [10]. Furthermore, CMA-ES attacks have succeeded where previous learning method, e.g., LR, has failed, for example, the Slender PUF [10].

5.2 Modeling Attack Test Setup

All the CMA-ES attacks are executed using MATLAB 2012b, where the CMS-ES code is adopted from [30], and the processor has an Intel i7-3770CPU@3.4GHz CPU and a 16GB memory.

5.2.1 OB-CRP Generation

Similar to studies in [8], [7], [23], [9], [10], the CRPs of underlying APUFs are generated through simulations that can effectively model a physical APUF architecture. The delay values for each stage in an APUF are randomly produced following a standard normal distribution. When a specific response is required, a given challenge $C$ is applied to the APUF to determine which delay values $\sigma_i^{0/1}$ in each stage will be selected according to the logic value of $c_i$. Then, according to the linear additive delay model, the selected delay values of two electrical signal paths are added up, separately, and compared with each other to generate a response bit. Artificially injected noise is not used in the following attack tests. Therefore, a purely 'logic' security of PUF design is evaluated. In this way, the evaluated security represents an upper bound on OB-PUF’s resilience to modeling attacks [8].
As for generation of an OB-CRP for OB-PUFs, first, a partial challenge \(C_{OB} \) is randomly generated. Next, one of \( p \) possible full length challenges \( C \) is formed according to a challenge obfuscation vector \( S_C \) determined by a specific randomly chosen pattern vector as illustrated in Fig.\( \ref{fig:fig5} \). Then its corresponding response \( R \) is produced by applying the formed full length challenge to the PUF block. As illustrated in Fig.\( \ref{fig:fig6} \), the obfuscated response \( R_{OB} \) is obtained by XORing a specific response obfuscation vector \( S_R \) with \( R \).

5.3.1 Response Hamming Distance based Fitness Metric

From the response Hamming distance based fitness metric is a key challenge in mounting ES attacks. We start with Becker’s work \([9, 10]\) to achieve ES attacks, they are based on: i) response Hamming distance, ii) response Hamming weight and iii) response reliability. Choosing a fitness function or metric is a key challenge in mounting ES attacks. We start from the response Hamming distance based fitness metric following Becker’s work in \([9]\).

5.3 CMA-ES Attacks on Reconfigurable OB-PUF

Though there are variants of ES algorithms, we follow Becker’s work \([9, 10]\) using covariance matrix adaptation ES (CMA-ES) \([30]\).

Overall, there are three fitness metrics in the literature \([9, 10]\) to achieve ES attacks, they are based on: i) response Hamming distance, ii) response Hamming weight and iii) response reliability. Choosing a fitness function or metric is a key challenge in mounting ES attacks. We start from the response Hamming distance based fitness metric following Becker’s work in \([9]\).

5.2.2 Criterion for Breaking OB-PUFs

In general, to successfully break a PUF, the prediction error rate of the learnt PUF model should be less than the \( \hat{p}_{intra} \) of the PUF. In other words, the prediction accuracy of the learnt model should be higher than \( 1 - \hat{p}_{intra} \). We refer to this minimum prediction accuracy as \( P_{min} \) and define it formally for OB-PUF below.

**Definition 7. Minimum prediction accuracy requirement.**

The \( P_{min} \) is the minimum prediction accuracy of the learnt PUF model that can be considered to successful break a PUF.

For OB-PUFs, based on \( \ref{eq:22} \), \( P_{min} = 1 - \hat{p}_{intra} \) then \( P_{min} \) is expressed as:

\[
P_{min} = \sum_{i=0}^{n_{initmatch}} \left( \frac{n_{ins}}{i} \right) \times (1 - \hat{p}^{intra})^{(n_{ins} - i)} \times (\hat{p}^{intra})^i,
\]

where \( \hat{p}^{intra} \) is the binomial probability estimator of intra-distance distribution of the underlying PUF, e.g., APUF, in the PUF block. In the following studies in Section 5, we use the worst-case \( \hat{p}^{intra} = 5\% \) obtained when an APUF is exposed to 45°C variation in temperature and 2% variation in supply voltage \([21]\).

Then the aim of an adversary is to build a model of an OB-PUF with prediction accuracy that is higher than the \( P_{min} \). If this is achieved, the OB-PUF is broken.

5.3.1 Response Hamming Distance based Fitness Metric

To ease understanding of evaluating the fitness of a candidate delay time vector, \( \omega \), when the reconfigurable pattern vectors are employed, we first describe how to define a fitness function under the assumption that pattern vectors are fixed and publicly known. For each OB-CRP, by knowing all pattern vectors, an adversary forms all possible full length challenges, \( C_{i1}, ..., C_{ip} \) and, all possible obfuscated responses \( R_{OBi1}, ..., R_{OBip} \) are computed according to the selected \( \omega \) and using all the known pattern vectors. Now, the minimum FHD between all possible computed obfuscated responses and the under-attack OB-PUF’s obfuscated response is evaluated:

\[
f_i = \min_{j=1,...,p} \{ f_{\text{HD}}(R'_{OBij}, R_{OB}) \}.
\]

Then the fitness metric \( f \) to evaluate the goodness of candidate, \( \omega \), is acquired by summing all \( f_i \) of all OB-CRPs; smaller the \( f \), fitter the learnt PUF model \([10]\).

Notably, the ES algorithm needs to iterate many generations before reaching the best delay time vectors \( \omega \). In each generation, many children are produced; for each child, it has a corresponding delay time vector inherited from its parent(s).

Now considering the implemented reconfigurable pattern vectors during authentication phase, in order to evaluate the fitness function in Eq \( \ref{eq:24} \), for each child in each generation, and for each authentication session, the ES algorithm has to first work out the pattern vector by predicting, then form all possible full length challenges, all based on a delay time vector of each child in every generation. At the end, all \( f \) for all sessions are summed up and normalized to form a fitness metric that is applicable to the reconfigurable OB-PUF.

The crux of reconfigurable pattern vector is the stringent requirement of predicting correct inserted values by an adversary. The full length challenge feature has a very high chance of being greatly different from the correct one if any bit of the inserted values is erroneous. This can be observed from Eq \( \ref{eq:5} \), where the challenge feature is a function of all bits in the challenge vector. It seems that arbitrarily measuring the OB-PUF by the adversary may not be a good choice to collect OB-CRPs even when the physical measurements on OB-PUF is allowed. Because, the adversary is unable to guarantee the \( p \times m \) inserted values being error freely generated when arbitrary partial challenges \( C_{OB} \) are applied. The \( p \times m \) error free inserted values are foundations for correctly evaluating the rest \( n \) OB-CRPs. If the inserted values, thus pattern vectors, are incorrect, then during the ES attack, the incorrect OB-CRP lead to misleading fitness assessments of candidate OB-PUF models.

In addition, the reconfiguring pattern vector makes the attack computation time significantly to be increased. With regarding to a fixed public known pattern vector as in our conference design \([5]\), the ES algorithm is only required to compute all possible full length challenges and then form their corresponding challenge features once. The same challenge features are applicable for each child in each generation. When the reconfigured pattern vector is implemented, such a one-time challenge features computation is no longer applicable. For each session, each child in each generation, the challenge features vary. Therefore, ES algorithm has to recompute the refreshed pattern vector first, then form all possible full length challenge per authentication session, then computes their corresponding challenge features. Such a re-computation of reconfigured pattern vectors and then challenge features per session is unavoidable for any child in any generation of the ES algorithm. This recomputation results into a significantly increased computation time of the ES attacks when the reconfiguring pattern vector per session is applied.
In the following, we evaluate the security of reconfigurable OB-PUFs under CMA-ES attacks.

**OB-PUF(2,2,0).** We first test OB-PUF(2,2,0), where two APUFs responses are XORed within the pattern vector reconfiguring block to produce pattern vectors per session. Attack results are shown in Table 3. Considering the $P_{\text{min}} = 90.25\%$ and the $P_{\text{pred}}$ is around 77%, the OB-PUF(2,2,0), hence, has not yet been broken. Recall that the $m$ is the eliminated bits in the full length challenge. The results indicate that a larger $m$ leads to a lower $P_{\text{pred}}$. This relies on the fact that the adversary’s model has a lower probability of correctly predicting all $p \times m$ inserted values all correct when the $m$ increases. Prediction errors of the computed $p \times m$ inserted values change the full length challenge feature and consequently increase erroneousness in the final computed obfuscated responses. This explains the increased $P_{\text{pred}}$ from 77.02% when $m = 3$ to 68.81% when $m = 8$. When a larger $m$ such as 6 is used, the prediction accuracy decreases greatly. We can observe this based on row1 and row5. From the ES learning perspective, the reason lies in the hardness of correctly determining all $p \times m$ inserted values, where an inaccurate determination.

In addition, under expectations, the computation time is significantly increased compared with attacking fixed pattern vectors that only costs around 15 minutes. When $N_{\text{session}} = 100$, $N_{\text{generation}} = 100$ and $n = 300$—this is the default setting when performing authentication as quantified in Table 2—the computation time is significantly prolonged to more than five hours as shown in row3.

**OB-PUF(4,4,0).** We now test OB-PUF(4,4,0). The test starts with using only a single APUF to produce reconfigured pattern vectors. In this context, the $P_{\text{pred}}$ is up to 76.42% as shown in row1 that is close but still lower than the $P_{\text{min}}$ of 81.45% according to Eq (23). When a XOR2-APUF within the pattern vector reconfiguring block is used, the $P_{\text{pred}}$ is significantly reduced to 56.87%, see row4. The reduced $P_{\text{pred}}$ is mainly attributed to the increased erroneousness that occurs during the determination of pattern vectors by the adversary’s model as a result of the XORing operation. On the other hand, this result agrees well with Becker’s view that an obfuscated PUF construction such as the Slender PUF or the OB-PUF in conjunction with PUFs already resistant to traditional ML attacks increases its resistance to CMA-ES attacks. The side product of increased computation time overhead is confirmed again by observing the last column.

In Fig. 10 (a), it depicts the progression of five runs of the CMA-ES on the OB-PUF(4,4,0). For all runs, $N_{\text{session}} = 100$, $n = 600$ and $m = 6$. We can see, for all runs, the $P_{\text{pred}}$ converges within 100 generations with the first 50 generations leading to an increased $P_{\text{pred}}$. After that, increasing the generations does not help improve the $P_{\text{pred}}$. This lies on the $p \times m$ inserted values, reconfigured per session, are hard to be computed all correctly by an adversary, which eventually misdirects the ES evaluation and disrupts further optimization. Each run is up to 400 generations and takes around 67 hours. The $P_{\text{pred}}$ is still far below the targeted $P_{\text{min}} = 81.45\%$, which validates the significantly resilience to CMA-ES attacks when the pattern vector reconfiguration is implemented. Fig. 10 (b) investigates the $P_{\text{pred}}$ when the $N_{\text{session}}$ or $n$ is increased, more specifically, more OB-CRPs are exploited for training. Using more OB-CRPs has no, at least negligible, improvement of adversary’s model prediction accuracy, which confirms one more time that reconfigurable latent pattern vectors misdirect the ES optimization.

We have extensively evaluated the OB-PUF’s security under the ES attacks using the response Hamming distance based fitness metric. Next we analyze the response Hamming weight and response reliability based fitness metrics, respectively.

### 5.3.2 Response Hamming Weight Based Fitness Metric

CMA-ES attacks using response Hamming weight fitness function have broken the security of the slender PUF. Though OB-PUF and Slender PUF both obfuscate the direct relationship between a challenge and its response, there is a major difference between the OB-PUF and Slender PUF that eliminates the OB-PUF’s vulnerability to CMA-ES attacks employing the Hamming weight based fitness test. In the Slender PUF protocol, each authentication session shares the same random indice that is, to be simplified, alike the random indice determining which pattern vector is chosen.

![Figure 10](image-url)
in the OB-PUF. But in OB-PUF, the random indice changes per OB-CRP rather than per session as in the Slender PUF protocol. In addition, to properly evaluate the Hamming weight based objective function, one premise is that the \( n_{\text{ins}} \) should be large enough in order to greatly outperform the noise effects (equivalent to a high signal to noise ratio, in short, SNR), which has been pointed out and also verified in [10]. Generally, to help evaluate the Hamming weight fitness performance, a \( n_{\text{ins}} \) that is in hundreds magnitude is necessary, however, the very small \( n_{\text{ins}} \) of OB-PUF is far way from performing the CMA-ES attacks based on the Hamming weight fitness that is applicable to the Slender PUF. Please also note that the Hamming weight of the obfuscated response \( R_{\text{OBF}} \) is already elaborately designed to be balanced, detailed in Section 5.3.3 this further eliminates the concern of Hamming weight fitness test enabled CMA-ES attacks.

5.3.3 Response Reliability based Fitness Metric
In [9], the reliability-based CMA-ES attack is applicable to XOR-APUFs. A proper reliability based fitness to mount CMA-ES attack on the OB-PUFs has not been figured out. If somehow, a proper reliability based fitness can be found for OB-PUF. This attack is still difficult to the reconfigurable OB-PUF. The main reason is that the correct determination of inserted values and then full length challenge feature is still required before performing such a reliability fitness based ES attacks. In other words, even the response of ‘1’/’0’ is unnecessary, which full length challenge presents an unreliable response is still required [10].

5.4 Discussion
It was recognized that employing APUFs as building blocks becomes very challenging in front of CMA-ES attacks [10] even that APUFs has extremely attractive properties such as compact structure and, most importantly, large CRP space. The reconfigurable OB-PUF serves an initial investigation of preventing powerful ES attacks. From the designing of reconfigurable OB-PUF, three hints are learned for future more efficient constructions: i) forcing the ES to compute such as the challenge feature in each child and in each generation, which exploits the uncircumventable process of the ES algorithm to significantly increase its computation time, ideally to be exponentially increased; ii) increasing the number of APUFs in a XOR-APUF, e.g., more than two, within the pattern vector reconfiguring block can significantly increase the OB-PUF’s ML-resilience further, this has been validated in Table 4. Or increasing the number of \( p \times m \) such as such as \( p = 4, m = 18 \) to exponentially decrease the capability of the adversary’s model predicting all pattern vectors all correct as validated in Table 3 and Table 4. iii) fully exploiting the uneven access to the underlying PUFs would be a plausible tool to exploit asymmetrical information obtained between the trusted party, e.g., the server and other party, e.g. adversary. For example, the server has the capability of generating error free responses on demand based on the learned statistical model of a basic APUF [29].

6 Conclusion
In this paper, we proposed a reconfigurable latent obfuscation technique that results into the design of reconfigurable OB-PUFs. This is a continuous investigation of a strong PUF scheme that enables the most desirable PUF-oriented lightweight entity authentication either using error correction logic nor a cryptographic algorithm, which has been keeping pursued for over a decade in the PUF community [22, 31].

The OB-PUF prevents an adversary carrying out successful modeling attacks due to that both pattern vectors and their selection to form possible full length challenges are unknown, but still allows a server to successfully authenticate an OB-PUF. In addition, pattern vectors are also reconfigured on demand and act as one-time pads per authentication session. We implement the most powerful modeling attacks to date, CMA-ES attack, to evaluate reconfigurable OB-PUFs security through extensive case studies, and demonstrate the significant modeling attack resilience. The reconfigurable OB-PUF studied in this paper serves as an very initial PUF designs that show significantly increased resilience to ES attacks without constraints on the available authentication rounds as in previous work [12]. We also summarize several useful hints to suggest future PUF designers to propose more efficient ES attacks resilient PUF constructions.

7 Acknowledgment
This research was supported by a grant from the Australian Research Council (DP14013448). The authors would also like to thank the sponsorship from China Scholarship Council (201306070017).

References
[1] B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, Silicon physical random functions, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pp. 148–160.
[2] R. Anderson, Security Engineering, John Wiley & Sons, 2008.
[3] G. E. Suh, S. Devadas, Physical unclonable functions for device authentication and secret key generation, in: Proceedings of the 44th Annual Design Automation Conference, ACM, 2007, pp. 9–14.
[4] S. Katzenbeisser, Ú. Kocabas, V. Rožič, A.-R. Sadeghi, I. Verbauwhede, C. Wachsmann, PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon, in: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, 2012, pp. 283–301.
[5] L. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, D. C. Ranasinghe, Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices, in: IEEE Percom Workshops, IEEE, 2016, DOI: 10.1109/PERCOMW.2016.7457162.
[6] D. Lim, Extracting secret keys from integrated circuits, Ph.D. thesis, Massachusetts Institute of Technology (2004).
[7] U. Rührmair, P. Sehnke, J. Söltzer, G. Dör, S. Devadas, J. Schmidthuber, Modeling attacks on physical unclonable functions, in: Proc. 17th ACM Conf. Computer and Communications Security, 2010, pp. 237–249.
[8] U. Rührmair, J. Söltzer, P. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dör, J. Schmidthuber, W. Burleson, S. Devadas, PUF modeling attacks on simulated and silicon data, IEEE Transactions on Information Forensics and Security 8 (11) (2013) 1876–1891.
[9] G. T. Becker, The gap between promise and reality: On the insecurity of XOR Arbiter PUFs, in: Cryptographic Hardware and Embedded Systems, Springer, 2015, pp. 535–555.
[10] G. T. Becker, On the pitfalls of using Arbiter-PUFs as building blocks, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34 (8) (2015) 1295–1307.
[11] M. Roel, Physically unclonable functions: Constructions, properties and applications, Ph.D. thesis, University of KU Leuven (2012).
[12] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, I. Verbauwhede, A lockdown technique to prevent machine learning on PUFs for lightweight authentication, IEEE Transactions on Multi-Scale Computing Systems.

[13] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, S. Devadas, Identification and authentication of integrated circuits, Concurrency and Computation: Practice and Experience 16 (11) (2004) 1077–1098.

[14] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, S. Devadas, A technique to build a secret key in integrated circuits for identification and authentication applications, in: Proc. IEEE Symp. VLSI Circuits, Digest of Technical Papers, 2004, pp. 176–179.

[15] L. Liu, S. Sreethara, D. K. Krishnappa, P. Shabadi, W. Burleson, Design and validation of arbiter-based PUFs for sub-45-nm low-power security applications, IEEE Trans. Inf. Forensics Security 7 (4) (2012) 1394–1403.

[16] Holcomb, Daniel E, W. P. Burleson, K. Fu, Power-up SRAM state as an identifying fingerprint and source of true random numbers, IEEE Transactions on Computers 58 (9) (2009) 1199–1210.

[17] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koerber, A.-R. Sadeghi, Remanence decay side-channel: The PUF case, IEEE Transactions on Information Forensics and Security 11 (6) (2015) 1106–1116.

[18] M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, J. Szefer, Run-time accessible DRAM PUFs in commodity devices, in: Cryptographic Hardware and Embedded Systems (CHES), Vol. 9813, Springer, 2016, p. 432.

[19] C. Herder, M.-D. Yu, F. Koushanfar, S. Devadas, Physical unclonable functions and applications: A tutorial, Proc. IEEE 102 (2014) 1126–1141.

[20] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, D. Abbott, Emerging physical uncloneable functions with nanotechnology, IEEE Access 4 (2016) 61–80.

[21] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk, S. Devadas, Extracting secret keys from integrated circuits, IEEE Trans. Very Large Scale Integr. (VLSI) Syst 13 (10) (2005) 1200–1205.

[22] M.-D. Yu, D. M’ Raihi, R. Sowell, S. Devadas, Lightweight and secure PUF key storage using limits of machine learning, in: Cryptographic Hardware and Embedded Systems, Springer, 2011, pp. 358–373.

[23] M. Majzoobi, F. Koushanfar, M. Potkonjak, Testing techniques for hardware security, in: IEEE International Test Conference – ITC, 2008, DOI:10.1109/TEST.2008.470636.

[24] M. Majzoobi, F. Koushanfar, Potkonjak, Lightweight secure PUFs, in: Proc. Int. Conf. Computer-Aided Design, 2008, pp. 670–673.

[25] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, S. Devadas, Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching, IEEE Transactions on Emerging Topics in Computing 2 (1) (2014) 37–49.

[26] M.-D. Yu, D. M’ Raihi, I. Verbauwhede, S. Devadas, A noise bifurcation architecture for linear additive physical functions, in: IEEE Int. Symp. Hardware-Oriented Security and Trust, 2014, pp. 124–129.

[27] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, S. Devadas, Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching, in: IEEE Symposium on Security and Privacy Workshops–(SPW), Vol. 6917, 2012, pp. 33–44.

[28] B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, Controlled physical random functions, in: 18th Annual Computer Security Applications Conference, IEEE, 2002, pp. 149–160.

[29] X. Xu, W. Burleson, D. E. Holcomb, Using statistical models to improve the reliability of delay-based pufs, in: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2016, pp. 547–552.

[30] N. Hansen, The CMA evolution strategy: A tutorial, arXiv preprint arXiv:1604.00772.

[31] J. Delvaux, R. Peeters, D. Gu, I. Verbauwhede, A survey on lightweight entity authentication with strong PUFs, ACM Computing Surveys (CSUR) 48 (2) (2015) 26.