FLAM-PUF: A Response–Feedback-Based Lightweight Anti-Machine-Learning-Attack PUF

Linjun Wu, Yupeng Hu, *Senior Member, IEEE*, Kehuan Zhang, *Member, IEEE*, Wenjia Li, *Senior Member, IEEE*, Xiaolin Xu, *Member, IEEE*, and Wanli Chang, *Member, IEEE*

***Abstract*—Physical unclonable functions (PUFs) have been adopted in many resource-constrained Internet of Things (IoT) applications to provide effective and lightweight solutions for device authentication. However, an attacker can collect challenge–response pairs (CRPs) of a strong PUF, to build a machine learning (ML) model and mimic its behavior, i.e., predicting the responses of unseen challenges with high accuracy. Although several PUFs have been proposed to resist such modeling attacks, they incur high hardware overhead. Developing a PUF primitive with low hardware cost and high resistance to ML attacks is thus a crucial task. In this article, we propose the first response–feedback-based lightweight anti-ML-attack PUF (FLAM-PUF). It is only composed of one arbiter PUF (APUF) and one Galois linear-feedback shift register (LFSR), with some basic logic gates, reducing more than 62% hardware cost compared with the state-of-the-art robust strong PUFs. Specifically, FLAM-PUF leverages a cost-effective feedback loop structure to dynamically control and update the LFSR configuration. FLAM-PUF has two main characteristics: 1) it feeds back a 1-bit response in every cycle to intentionally poison the data of the CRP set for training. To resist ML-based modeling attacks, the 1-bit response can randomly update one coefficient of the feedback polynomial to implant more complex correlations into the model built by attackers and 2) it takes advantage of an $n-$bit response feedback-controlled reconfigurable Galois LFSR to enlarge the original challenge space of the APUF. Extensive experimental results show that the proposed FLAM-PUF achieves near-optimal uniformity, uniqueness, and reliability. Our scheme works well under standard attack models with public crucial initial information. In particular, the prediction accuracy of modeling attacks against FLAM-PUF is nearly 50% under the four widely used ML algorithms, i.e., support vector machines (SVMs), logistic regression (LR), covariance matrix adaptation evolution strategy (CMA-ES), and deep neural networks (DNNs), indicating excellent resistance against these ML attacks.**

***Index Terms*—Lightweight, machine learning (ML) attacks, physical unclonable function (PUF), reconfigurable linear-feedback shift register (LFSR), response feedback.**

Manuscript received 14 July 2022; accepted 26 July 2022. Date of current version 24 October 2022. This work was supported in part by the Hunan Province Outstanding Youth Fund Project under Grant 2022JJ10018; in part by the National Natural Science Foundation of China under Grant 61872130; and in part by the Hong Kong, SAR, Research Grants Council (RGC) General Research Fund under Grant 14208818. This article was presented in the International Conference on 2022 and appears as part of the ESWEEK-TCAD special issue. This article was recommended by Associate Editor A. K. Coskun. *(Corresponding authors: Yupeng Hu; Wanli Chang.)*

Linjun Wu and Yupeng Hu are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410012, Hunan, China (e-mail: wulinjun777@hnu.edu.cn; yphu@hnu.edu.cn).

Kehuan Zhang is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: khzhang@ie.cuhk.edu.hk).

Wenjia Li is with the Department of Computer Science, New York Institute of Technology, New York, NY 10023 USA (e-mail: wli20@nyit.edu).

Xiaolin Xu is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: x.xu@northeastern.edu).

Wanli Chang is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410012, Hunan, China, and also with the Central Software Institute, Huawei Technologies, Shenzhen 518129, China (e-mail: wanli.chang.rts@gmail.com).

Digital Object Identifier 10.1109/TCAD.2022.3197696

I. Introduction

Manufacturers and designers did not prioritize security issues during the early years of the Internet of Things (IoT) design and applications. Nowadays, various IoT services, which are designed for industrial automation [1], smart home [2], traffic safety [3], e-health [4], etc., are exposed to cyber threats [5]. These security issues are becoming a bottleneck limiting the development of IoT, especially the low-end and low-cost IoT devices with minimal or no security solutions, such as sensors and radio-frequency identification (RFID) tags [6], [7]. Consequently, hackers may compromise the IoT devices to launch attacks. For example, around 150 000 infected IoT devices caused a massive attack [8] in 2016. These infected IoT devices involve surveillance cameras, routers, and other intelligent devices. The combination of computing and networking power of thousands of IoT devices makes the IoT environment a potentially deadly cyber threat [9].

State-of-the-art cryptographic-based security mechanisms introduce high computational complexity and high cost to store secret keys. While most IoT node devices have limited central processing unit (CPU), memory, and battery power resources, it is challenging for them to afford the high hardware and power consumption of complex cryptographic algorithms, key stores, and protection mechanisms. Therefore, it is imperative to seek a cost-effective system security solution. Physical unclonable function (PUF) is a promising hardware security primitive that provides a lightweight solution for key generation, intellectual property (IP) licensing, key sharing, and hardware metering. A PUF generates a unique input-output mapping relationship for each device leveraging the inherent random process variations from manufacturing. Specifically, these input and outputs are called challenge–response pairs (CRPs). Ideally, a PUF instance should be unclonable and unpredictable thanks to its disordered and complex structure. According to the security properties of CRPs, PUFs can be classified into strong PUFs and weak PUFs. The weak PUF only has a small CRPs space and is mostly used for key generation [10]. In contrast, the strong PUF, like arbiter PUF (APUF) [11], has a huge CRPs space (e.g., exponential numbers of CRPs) and is used in device authentication and identity recognition. It is expected that even given access to the PUF instance, an adversary still cannot predict the unseen CRPs. Moreover, the strong PUFs will discard the used CRPs from the available CRPs pool to protect against man-in-the-middle and replay attacks [12].

Strong PUFs are subject to severe security issues due to the emergence of machine learning (ML)-based modeling attacks. In a modeling attack, an adversary builds a generic model with polynomial parameters for a class of PUFs to simulate the CRP mapping relationship. If the model is linear, the adversary can simply apply ML algorithms to estimate the model parameters by observing the obtained CRP set. It is infeasible to build an accurate model of the strong PUF that uses only a subset of CRPs to "train" the model to learn its complete CRP behavior. However, once the model is able to predict the response of the PUF with an error probability lower than the maximum environmental variation, the adversary can impersonate the original PUF with the model successfully [13].

ML technique is a natural and powerful tool to train models for modeling attacks. The attacker divides the obtained CRP subset into a training set and a testing set. In each iteration, the ML algorithms input challenge into the established PUF mathematical model and calculate the corresponding response. Then, the algorithm calculates the loss function based on the difference between the response output of the model and the actual value. The model parameters are updated according to a specific strategy (the gradient descent for the logistic regression (LR) algorithm and the maximum likelihood for the support vector machines (SVMs) algorithm in this article). When the iteration termination condition is satisfied, the training process ends. The degree of difference between the model trained by the algorithm and the real PUF mapping can be verified by the prediction accuracy of the testing set. The experimental results in [14] show that the modeling accuracy is over 95% in 0.01 s by collecting 650 pairs of CRPs for a 64-stage APUF [11]. It is easy for an attacker to obtain CRPs due to the lack of a protection mechanism to restrict access to CRPs. Once an accurate model is built successfully with the CRPs of a PUF, the protocols built on it are vulnerable.

Motivated thereby, a plethora of efforts have been devoted recently to improving the robustness of strong PUFs against ML attacks. The existing high-security strong PUF solutions still suffer from high complexity and hardware overhead [15], [16], [17]. Nevertheless, most of these PUF structures can still be modeled successfully by various attack methods [11]. Linear-feedback shift register (LFSR) is a high-performance and low-overhead sequential circuit that can generate repeatable pseudo-random sequences. It has found important utilization in traditional information security, such as stream encryption and cyclic redundancy check (CRC). Combining LFSR with classical PUFs turns out to be a worthy research direction. In existing works, LFSRs with fixed feedback coefficients were often used for expanding the challenge space [18], [19] or CRPs confusion [20]. For instance, CRC PUF [21] is the first work to make CRPs confused by randomly updating feedback polynomial $g(x)$ for LFSR, without detailing how the feedback polynomial is updated; SRPUF [22] needs $(n + k)$ APUFs to update the feedback coefficient, leading to an unaffordable hardware overhead. Therefore, developing a PUF primitive with low hardware overhead but high resistance to ML attacks has become a primary challenge in recent years.

Our contributions to this work are as follows.

Response Feedback Loop-Based Resistance to ML: This work proposes the first response–feedback-based lightweight anti-ML-attack PUF (FLAM-PUF). A FLAM-PUF is composed of an APUF and a reconfigurable Galois LFSR in a loop structure. The reconfigurable LFSR updated by the response feedback of APUF is the central part of the obfuscation method in this article. Unlike existing configurable LFSRs, our LFSR generates the direct challenge C^* for the APUF, then a 1-bit response r^* of the APUF will dynamically be fed back to the reconfigurable LFSR, to randomly update its polynomial in each cycle. Subsequently, FLAM-PUF can intentionally poison the data of CRP set for training by implanting problematic correlations into the target model of attackers, improving the obfuscation and resistance to ML modeling attacks. Moreover, FLAM-PUF exploits a n-bit response feedback-controlled reconfigurable Galois LFSR to enlarge the original challenge space of the APUF.

Lightweight Architecture: Instead of comprising multiple strong PUF combinations or complex encryption circuits, FLAM-PUF is lightweight and only consists of an APUF, a Galois LFSR, and some basic logic gates. Furthermore, we did not need any extra secret key information, and thus there is no additional memory overhead. Compared to state-of-the-art robust strong PUFs, FLAM-PUF has a lightweight hardware architecture, which can reduce more than 62% hardware area in the 64 stage and 128 stage.

Public Circuits Structure: The circuit structure of FLAM-PUF can be public as follows. First, we assume that the attackers know some crucial initial information, including the circuit structure and the feedback polynomial initial state. Second, we can leak out the *specific position of feedback point in circuit structure to attackers* to demonstrate the security of our obfuscation, as we show in Sections V-D and V-E. Third, the attackers can eavesdrop on the part of the CRPs from the unreliable channel.

Extensive Experiments: The experimental results demonstrate that the proposed reconfigurable solution can achieve a prediction accuracy of nearly 50% with four well-known ML algorithms, equivalent to a random guess. Additionally, the uniqueness, uniformity, and reliability of the proposed PUF are close to the ideal value.

The remainder of this article is organized as follows. Related works and technical challenges are reviewed in Section II. Section III introduces the preliminaries of classic PUF models, LFSR, and four widely used modeling attack algorithms. Section IV presents the structure of the proposed FLAM-PUF. Section V elaborates on the comprehensive evaluations of performance evaluations and security for FLAM-PUF. Finally, Section VI concludes this article and points out future research directions.

II. Related Work and Challenges

Strong PUFs are subject to various ML modeling attacks [13], [14], [23]. The main threat model in these works is that an attacker can build a generic model to approximate the input-output mapping of a given PUF, and then use the eavesdropped CRPs set to estimate the model parameters for a specific PUF instance. The ML algorithms that improve the efficiency of estimating parameters have undermined the security of strong PUF. For example, in the APUF circuits, there is a linear relationship between the accumulative path delay and the challenge vectors. The additive linear delay model can accurately describe the functionality of APUF. Recent years have seen considerable defense technologies for improving the anti-ML-attack abilities of strong PUFs.

A. Related Work of Defense Technologies

Existing anti-ML-attack solutions could broadly be divided into two categories as follows.

Strong PUF Combination and Confusion: These solutions combine the classic strong PUFs (e.g., APUF) with other function modules to strengthen the whole security of the circuit structure. The most representative XOR APUF [24] can obtain a global response from parallel APUFs by XORing the original responses. Ising PUF [15] applies each APUF spin to the Ising model, and the interaction between the spins produces a global response. Interpose PUF [25] contains two XOR APUFs, i.e., n-stage x-XOR APUF and $(n+1)$-stage y-XOR APUF. The 1-bit response of x-XOR APUF is inserted into the original-bit challenge to form the $(n+1)$th bit challenge, which is provided to the y-XOR APUF to produce a global response. Wang *et al.* [26] proposed a dual-mode PUF that utilizes the parity of the number of inverters to resist modeling attacks. Configurable tristate (CT) PUF [17] can flexibly perform as an APUF, a ring oscillator (RO) PUF, or a bistable ring (BR) PUF [27], through a bitwise XOR-based mechanism obfuscating the challenge and response relationship. Controlled PUF [28], PUF-FSM [29], and OB-PUF [30] all use a hash module for logical obfuscation. In RPUF [31], CRC-PUF [21], and SR-PUF [22], an LFSR module undertakes a crucial role in CRPs confusing. In [32], an inverter is activated under certain conditions and reverses the response. A reverse response bit serves as poison data that interferes with attacker modeling. Zhang *et al.* [33] increased the difficulty of modeling by limiting the number of CRPs leaked out to attackers.

Nonlinear Sequential Circuits: In addition to reinforcing the classic strong PUFs, researchers are exploring a variety of nonlinear circuit characteristics, such as glitches [34] and voltage transfer characteristics (TVCs) [35]. Kumar and Burleson [36] and He *et al.* [37] created new CRP mapping based on the strong nonlinearity between the deviation of the symmetrical structure and the circuit characteristics. Various emerging micro–nano devices have brought forth new technologies for strong PUF, such as resistive random access memory (ReRAM) [38] and spin-transfer torque magnetoresistive random access memory (STT-MRAM) [39]. In [16], to produce CRPs, the PUF array is organized as two coupled inverter-based entropy sources cascaded via nonlinear Sbox transformations. The work in [40] exploits the sequence-dependent behavior to expand the CRP space for a memory PUF. The subthreshold operating state of complementary metal–oxide–semiconductor (CMOS) devices has the advantages of lower voltage, lower power consumption, and stronger nonlinear compared to its saturated operating state. Recently, some schemes employ the subthreshold current array as a basic PUF element to realize a highly nonlinear operation characteristic to resist ML attacks [41], [42], [43].

B. Technical Challenges

The primary technical challenge of the state-of-the-art obfuscation for resisting ML attacks is the excessive hardware overhead when the circuits confuse a CRP mapping with sufficient complexity to increase the difficulty of modeling attacks. A linear model easily simulates the properties of a single simple nonlinear circuit. Currently, more researchers are working on confounding CRPs by building a complex nonlinear system composed of multiple functional circuits. There must be enough circuit modules to successfully achieve the expected complexity and defend against ML modeling attacks. For instance, the number of APUFs in XOR-APUF cannot be smaller than 7. The combination of multiple circuits will inevitably cause more hardware overhead. Awano and Sato [15], Suresh *et al.* [16], Zhang *et al.* [17], and Rührmair *et al.* [44] combined some classic PUF circuits to increase the complexity of CRPs mapping while incurring high hardware costs.

On the other hand, APUF is a classic strong PUF, and its challenge–response space size is exponentially related to the number of switch modules. Compared with other PUFs, APUF can generate many CRPs with lower hardware resources, thus realizing key generation at a lower cost. However, APUF is subject to modeling attacks [45]. To resist modeling attacks, a series of strong PUFs are designed based on APUF like XOR APUF [24], feed-forward APUF [46], lightweight secure PUF [47], and interpose PUF [25]. Similarly, LFSR can efficiently generate pseudo-random sequences via simple operations and plays an important role in cryptography. The configurable LFSR dynamically updates the correlations between inputs and outputs with strong nonlinearity and randomness. Thereby LFSR can avoid cryptanalysis attacks by merely adding some simple logic gates (e.g., AND, NOT, and NAND). In this way, the LFSR enables the high security and low overhead tradeoff in lightweight application scenarios. Therefore, we employ the APUF and reconfigurable LFSR to build a response feedback-based lightweight anti-ML-attack strong PUF.

III. Preliminaries

This section introduces the preliminaries of classic PUF models, LFSR, and modeling attack algorithms. Table I lists some important parameters and terminologies used here.

TABLE I
LIST OF PARAMETERS AND TERMINOLOGIES IN THIS ARTICLE

Symbol	Description
Δ	the total time delay difference of APUF
w	the time delay feature vector
Φ	the feature vector
C	n-bit original challenge
C^*	n-bit direct challenge of APUF
r^*	one-bit direct response of APUF
R^*	n-bit direct response set of APUF
r	one-bit final response
R	n-bit final response set
a_j	the j-th register in LFSR
s_j^i	the state of j-th register in LFSR during i-th clock cycle
S^i	the registers state set of the LFSR during i-th clock cycle
G	the feedback coefficient set of LFSR
T	the transition function of LFSR

A. Classic PUF Descriptions and Models

Arbiter PUF: The APUF circuits consist of a sequence of multiplexer chains. The same signal propagates through two paths in parallel (top and bottom) to generate 1-bit response output. In a parallel or cross mode, the connection of each path is determined by the challenge bit provided to the two multiplexers at each stage. Ideally, the delays of two paths giving the same challenge should be the same. However, the actual delays of the two paths will be slightly different owing to process deviations. At the end of the chain, a latch is utilized as an arbiter to compare the delay difference between the top and bottom paths.

The additive linear delay model has become a standard description for the functionality of APUFs [14], [48] [23]. This model describes the total time delay difference between the top and bottom paths. As shown in Table I, $\Delta = \vec{w}\vec{\Phi}$ is the sum of the delays in the stages, where the time delay feature vector $\vec{w} = (w^1, w^2, \ldots, w^n, w^{n+1})^T$. $\delta_i^{0/1}$ denotes the time delay in stage i for the crossed (δ_i^1) or in parallel (δ_i^0) signal path, respectively. Then, $w^1 = ([\delta_1^0 - \delta_1^1]/2)$, $w^i = ([\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1]/2)$ for all $i = 2, \ldots, n$, and $w^{n+1} = (\delta_n^0 + \delta_n^1/2)$. The feature vector $\vec{\Phi}$ is the function of the applied n-bit challenge $C = (c_1, c_2, \ldots, c_n)$. Namely, $\vec{\Phi}(\vec{C}) = (\Phi_1(\vec{C}), \ldots, \Phi_k(\vec{C}), 1)^T$, where $\vec{\Phi}_l(\vec{C}) = \prod_{i=l}^{n}(1 - 2c_i)$ for $i \in [1, k]$. It is worth noting that for the convenience of modeling, the original challenge or response of value "0" is reprocessed as "−1." The response r of an APUF is defined as follows:

$$r = \mathrm{Sgn}(\Delta) = \mathrm{Sgn}\left(\vec{w}\vec{\Phi}\right). \quad (1)$$

XOR *Arbiter PUF:* XOR APUF [14], [24] employs k individual n-stage APUFs in parallel and provides the same challenge C to all of them. Each APUF generates its individual response r_i and then XORs the r_i to produce a global response. Assuming $r_i \in \{-1, 1\}$ as mentioned above, then we have $r_{\mathrm{XOR}} = \prod_{i=1}^{k} r_i$. According to (1), the parametric model of a k-XOR APUF is

$$t_{\mathrm{XOR}} = \prod_{i=1}^{k}\mathrm{Sgn}\left(\vec{\omega}_i^T\vec{\Phi}_i\right) = \mathrm{Sgn}\left(\prod_{i=1}^{k}\vec{\omega}_i^T\vec{\Phi}_i\right) \quad (2)$$

$$= \mathrm{Sgn}\left(\bigotimes_{i=1}^{k}\vec{\omega}_i^T\bigotimes_{i=1}^{k}\vec{\Phi}_i\right) = \mathrm{Sgn}\left(\vec{\omega}_{\mathrm{XOR}}^T\vec{\Phi}_{\mathrm{XOR}}\right). \quad (3)$$

MPUF and Logical Approximation Model: An MPUF [11] consists of (2^k+k) APUFs and a $(2^k : 1)$ multiplexer (MUX). It provides the same challenge C to all of the (2^k+k) APUFs. The responses $(r_0^d, r_1^d, \ldots, r_{2^k-1}^d)$ generated from the 2^k APUFs will be sent to the $(2^k : 1)$ multiplexer in parallel. The other k APUFs' responses $(r_0^s, r_1^s, \ldots, r_{k-1}^s)$ are responsible for selecting one response in $(r_0^d, r_1^d, \ldots, r_{2^k-1}^d)$ as the global response. In the logical approximation model, the $(2^k : 1)$ multiplexer can be decomposed into several 2 : 1 MUXs. Each 2 : 1 MUX can be implemented by the basic logical operations (NOT, AND, and OR), which can be approximated by the following functions:

$$\mathrm{NOT}\, a = 1 - a \quad (4)$$

$$a\,\mathrm{AND}\, b \approx \sigma(20a + 20b - 30) \quad (5)$$

$$a\,\mathrm{OR}\, b \approx \sigma(20a + 20b - 10). \quad (6)$$

Finally, the global response output the O of MPUF with k selection APUFs, which can be modeled as

$$O \approx \sum_{i=0}^{2^k-1}\left(r_i \times \mathrm{sinc}\left(i - \sum_{j=0}^{k-1}2^j \times r_j\right)\right) \quad (7)$$

where $\sigma(x)$ and $\mathrm{sinc}(x)$ are common activation functions in ML algorithms.

B. Linear-Feedback Shift Register

The input bit of an LFSR is the output of a linear function of two or more for its previous states (taps). There are two major schemes for connecting taps: 1) Fibonacci and 2) Galois. Here, we only focus on the Galois LFSR form. Let s_j $(j \in [0, n-1])$ represent the output state of each register and g_j $(i \in [0, n])$ is the feedback coefficient of LFSR, which controls whether the output of the final register s_{n-1} participates in the feedback operation (XOR). If $g_j = 1$, the input of the next register is $s_j \oplus s_{n-1}$; otherwise, $g_j = 0$, s_j does not participate in the feedback operation and directly enters the following register. It is worth noting that g_n and g_0 must be 1 to ensure that LFSR is dynamic and effective.

The register state of Galois LFSR is updated in every clock cycle by the state transition function $T : GF(2^n) \to GF(2^n)$. T^i indicates that the state of the registers has been updated i times. Let $S^i (i \in [1, +\infty])$ represent the register state of the LFSR in ith cycle and S^0 represent the initial state, and then S^i can be defined as follows:

$$S^i = T^i\left(S^0\right). \quad (8)$$

The updating of n registers in the ith clock cycle is described as follows:

$$\begin{pmatrix} s_0^i \\ \cdots \\ s_j^i \\ \cdots \\ s_{n-1}^i \end{pmatrix} \to \begin{pmatrix} s_{n-1}^{i-1} \\ \cdots \\ t\left(s_{j-1}^{i-1}\right) \\ \cdots \\ t\left(s_{n-2}^{i-1}\right) \end{pmatrix} \quad (9)$$

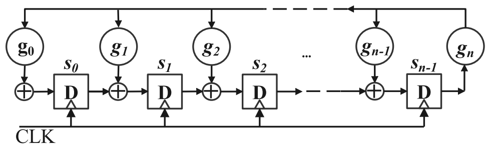

Fig. 1. Galois LFSR structure.

where s_j^i defines the state of the jth register in the ith cycle. Let $\oplus$ denote addition operation in GF(2), the function t is defined as follows:

$$t(s_j^i) = s_{n-1}^{i-1} \times g_j \oplus s_{j-1}^{i-1}. \tag{10}$$

With the development of cryptography, the security of traditional LFSR is not optimistic. Attackers can collect enough output sequences to calculate its structure. Therefore, the researchers proposed configurable LFSRs, which realize the dynamic configuration of LFSR parameters through different circuit structures. The configurable parameters involved in the previous works include: 1) the number of stages in the LFSR; 2) the number of taps in the feedback path; and 3) each tap position in the shift register stage [49], [50], [51]. In [21] and [22], it is adopted the 2-input AND gates to realize the configurable feedback function of LFSR.

C. Modeling Attack Algorithms

SVM Attacks: For each given CRPs train data set, attackers create a PUF model and use SVMs [13] to learn the CRPs train set. Challenges are the feature vector, and corresponding responses are as the label. After training, the SVM generates a learning model and is applied to learn a CRPs test set. Then, for each PUF, the learning results are compared with the actual results and calculate the correct rate as the prediction accuracy.

LR Attacks: LR is a widely used supervised ML algorithm in PUF attacks. In [14], the authors showed the LR attacking some strong PUFs like XOR APUF turns out to be significantly better than SVM and evolution strategy (ES). The LR algorithm uses the sigmoid function to convert the input data into probabilities and determines the label to which the feature vector belongs based on the probability values. In the application of the PUF attack, each n bit challenge C is used as the n-dimensional feature vector of LR. The response is the label judged according to the predicted probability result, and the time delay feature vector corresponds to the parameter of the maximum likelihood function to be solved. Whether the feature vector and the label have a linear relationship significantly impacts the quality of the prediction result. Therefore, the premise of an effective LR attack against PUF is that the PUF has a linear structure.

CMA-ES Attacks: Covariance matrix adaptation evolution strategy (CMA-ES) is a stochastic method for real-parameter optimization of nonlinear, nonconvex function [52]. The CMA-ES algorithm uses the parent's delay vector and some random modifications to generate the children's delay vector in each iteration. Then the algorithm selected the fittest instances from these child instances and kept them for the next generation as parents. In the experiment, we use the default parameters in [53].

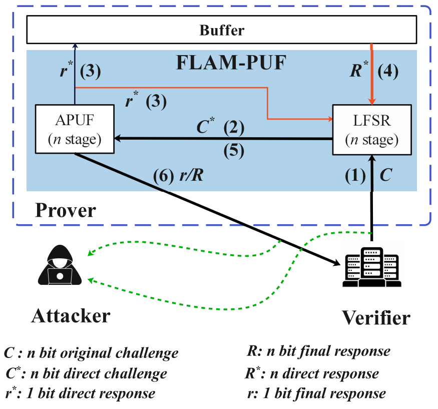

Fig. 2. FLAM-PUF structure and data obfuscation flow. The thin line between the modules in the figure represents a 1-bit signal, and the bold line represents a multibit signal (e.g., n bits). There are two steps (3) since the 1-bit response generated by APUF will be fed back to LFSR and buffer for temporary storage simultaneously.

DNN Attacks: Deep neural networks (DNN) construct a nonlinear model with a multilayer neural network and a nonlinear activation function. It is a powerful black box tool to solve different complex tasks in artificial intelligence.

IV. Proposed Method: Response Feedback-Based Lightweight PUF

Neither PUF combinations nor nonlinear circuits are cost effective in terms of their hardware implementations. To overcome the challenge brought by the high cost of hardware implementation, we leverage a cost-effective response feedback loop structure to control and update the LFSR configuration in every cycle dynamically.

A. Structure Overview

Fig. 2 shows the structure overview of the proposed FLAM-PUF, an APUF combined with a reconfigurable LFSR can form a close loop structure for CRPs confusion. The critical characteristics of FLAM-PUF are as follows. First, FLAM-PUF feeds back a 1-bit response in every cycle to intentionally poison the data of the CRP set for training. Notably, the 1-bit response can randomly update one coefficient of feedback polynomial to implant problematic correlations into the model to be built by attackers to resist ML-based modeling attacks. Second, a FLAM-PUF exploits an n-bit response feedback-controlled reconfigurable Galois LFSR to enlarge the original challenge space of the APUF. In this way, FLAM-PUF can combine circuit obfuscation with timing obfuscation at the cost of a simple loop circuit structure that obtains a significant degree of CRP obfuscation.

Based on the LFSR and APUF, the data obfuscation flow concludes the following steps, as also illustrated in Fig. 2.

Step (1): LFSR receives and confuses the n-bit original challenge C first.

Step (2): Then, LFSR produces a direct challenge C^* for the APUF. This step expands an original challenge C into a

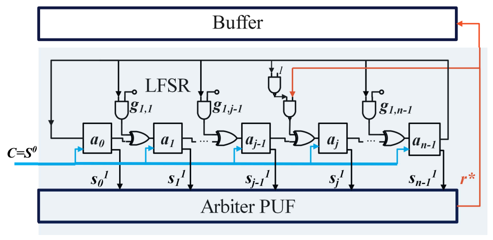

Fig. 3. First confusion and APUF.

series of direct challenges C^*. The attacker can only obtain the original challenge C and not the direct challenge C^*.

Step (3): In each cycle, APUF generates a 1-bit direct response r^* and transmits it simultaneously to LFSR for configuration updating and cache buffer. The 1-bit direct response r^* output by the APUF can be nearly equally likely to be 0 or 1. There will also be two equal possibilities for the following direct challenge generated by the LFSR aliasing affected by the feedback r^*.

Step (4): After repeating steps (2) and (3) for n times, the buffer collects a $(n-1)$-bit response $R^* = (r_1^*, r_2^*, \ldots, r_{(n-1)}^*)$ and feed them back to LFSR as the feedback coefficient for the secondary confusion. The 1-bit direct response r^* generated per cycle will cause (1/2) uncertainty in the system. At this step, $(n-1)$ bits of direct responses R^* have been generated, bringing 2^{n-1} uncertainties to our system.

Step (5): LFSR repeats the same operation of step (2) based on the input R^*, and then generates a new C^*.

Step (6): FLAM-PUF will generate the final response r at the kth cycle. It can also output an n-bit R from the kth to the $(k+n-1)$th cycle to enhance the obfuscation further. The entire PUF structure hides the real CRPs of APUF, which significantly increases the modeling difficulty for attackers.

Among the above six steps, steps (3) and (4) return the APUF responses to the LFSR to update configuration, which is critical to ensure the dynamic update of the entire design with high randomness, complexity, and security.

B. Initialization and First Confusion

In our scheme, the LFSR circuit plays a critical role in obfuscation. In each cycle, the LFSR produces a new challenge for the APUF and receives 1-bit response feedback from the APUF. The FLAM-PUF does not generate the final response r immediately in the first $(n-1)$ cycles, i.e., the first confusion, as shown in Fig. 3. During the first confusion, the feedback coefficient set G_1 is fixed. Only one feedback coefficient can be affected in each run since only a 1-bit response is fed back to implant problematic correlations into the target model of attackers. Specifically, the 1-bit response loaded in specific registers is responsible for poisoning the data of the CRP set for training by randomly updating one coefficient of feedback polynomial.

Challenges of APUF: The n-bit original challenge $C = (c_0, c_1, \ldots, c_{n-1})$ is first loaded into LFSR as its initial state S^0, $C = S^0$. The feedback coefficients $G_1 = (g_{1,1}, \ldots, g_{1,n-1})$ for the first confusion control the registers input via 2-input AND gates. As mentioned in Section III-B, to ensure the effectiveness of LFSR, g_n and g_0 should always be set to 1. Therefore, our method just needs $(n-1)$ feedback coefficient parameters. In each cycle, the n registers in LFSR $\{a_0, a_1, \ldots, a_{n-1}\}$ will generate an n-bit output as a direct challenge C^* of APUF. For instance, the first direct challenge is $C_1^* = S^1 = T^1(S^0) = T^1(C)$, where $T^1()$ is the transition function of LFSR in the first cycle. Similarly, we have

$$C_i^* = S^i = T^i\left(S^0\right) = T^i(C). \tag{11}$$

From (11), we can get the ith direct challenge C_i^* by obfuscating the original challenge C via LFSR's i runs.

One-Bit Response Feedback: For each direct challenge C^*, the APUF generates the corresponding 1-bit response r^* and sends it to the buffer. At the same time, r^* is fed back to the XOR gate before the jth register a_j in LFSR, and affects the corresponding output s_j^1. In Fig. 3, we replace a 2-input AND with two 2-input NAND gates as the feedback module. If $r^* = 0$, $s_{i,j} = \neg s_{i-1,j-1}$, thus r^* flips the state of s_j^i directly; otherwise, $r^* = 1$, $s_j^i = s_{j-1}^{i-1} \oplus s_{n-1}^{i-1}$, leading to $g_{1,j} = 1$. Based on (1) and the sign function Sgn(x), we can obtain

$$\begin{aligned} r_i^* &= \mathrm{Sgn}\left(w_1\Phi_1\left(S^i\right) + w_2\Phi_2\left(S^i\right) + \cdots + w_n\Phi_n\left(S^i\right)\right) + w_{n+1} \\ &= \mathrm{Sgn}\left(\vec{w}\vec{\Phi}\left(T^i(C)\right)\right) \end{aligned} \tag{12}$$

and r_0^* is directly generated from the original challenge C

$$r_0^* = \mathrm{Sgn}\left(\vec{w}\vec{\Phi}(C)\right). \tag{13}$$

Notably, Fig. 4 abstracts the workflow of response feedback in the ith and $(i+1)$th cycle during the first confusion. We assume that in the initial feedback coefficient set G_1, only $g_{1,1}$ equals 1; the others are 0 in this case. The feedback position is between the second and the third register, thus the ith direct response r_i^* will influence the third register's state.

C. Secondary Confusion and Final Response

To strengthen the obfuscation, we will perform secondary obfuscation, as shown in Fig. 5. After $(n-1)$ cycles, the buffer can collect and assemble an $(n$-1)-bit direct response $R^* = (r_1^*, r_2^*, \ldots, r_{n-1}^*)$, which is employed to update the feedback coefficient parameters of LFSR for the secondary confusion, namely, $G_2 = (r_1^*, r_2^*, \ldots, r_{n-1}^*)$. The initial state of the LFSR for the secondary confusion is S^{n-1}, which is the state set at the $(n-1)$th cycle. Therefore the FLAM-PUF can generate 1-bit final response r at any kth cycle for $k \geq n$ (e,g., $k = n$). According to (11) and (12), we can get

$$r = r_k^* = \mathrm{Sgn}\left(\vec{w}\vec{\Phi}\left(T^k(C)\right)\right). \tag{14}$$

To improve the authentication efficiency, we can generate an n-bit response from kth cycle to $(k+n-1)$th cycle as a final response R. In this case, the n-bit final response is $R = (r_k^*, r_{k+1}^*, \ldots, r_{k+n-1}^*)$. Obviously, the more obfuscation runs, the better obfuscation effect we can obtain at the cost of a longer delay.

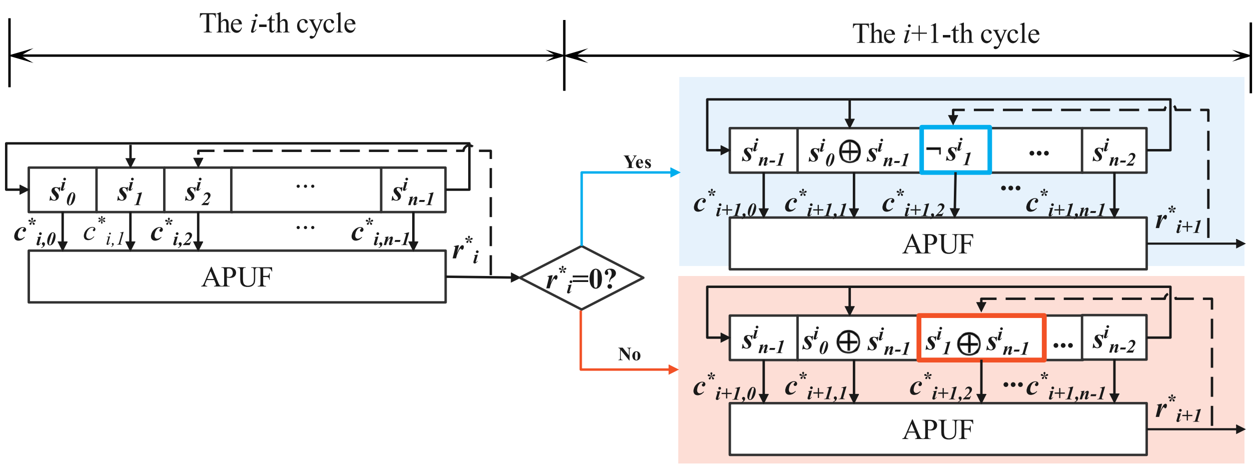

Fig. 4. Workflow diagram of response feedback in the in the ith and $(i+1)$th cycles.

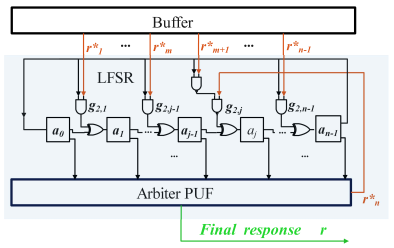

Fig. 5. Secondary confusion.

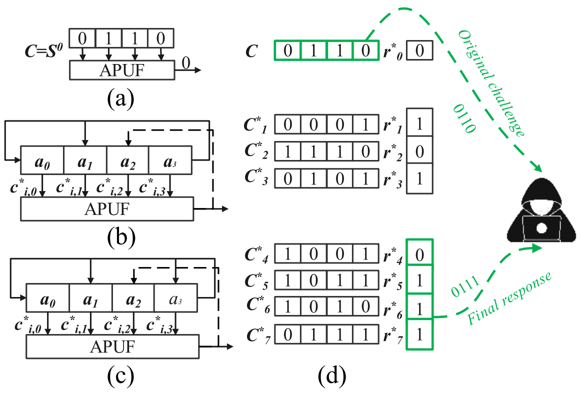

Fig. 6. Case study. (a) Initialization. (b) First confusion. (c) Second confusion. (d) CRP of APUF.

D. Case Study

Fig. 6 showcases the efficiency of the proposed method with a 4-stage APUF and an LFSR as follows.

1) *Initialization:* If the original challenge $C = S^0 = (s_0^0, s_1^0, s_2^0, s_3^0) = (0110)$ is loaded into the LFSR as the initial state, the initial feedback coefficient set is $G_1 = (100)$, and the feedback position is between the second and the third register (i.e., a_1 and a_2). Also, we assume $r_0^* = 0$.

2) *First Confusion:* LFSR generates the first direct challenge $C_1^* = S^1 = (s_0^1, s_1^1, s_2^1, s_3^1)$, where $s_0^1 = s_3^0 = 0$, $s_1^1 = s_0^0 \oplus s_3^0 = 0$, $s_2^1 = \neg s_1^0 = 0$, $s_3^1 = s_2^0 = 1$. Namely, C_1^*=(0001), and r_1^*=1. Similarly, we can obtain $C_2^* = (1110)$, $r_2^* = 0$, and $C_3^* = (0101)$, $r_3^* = 1$.

3) *Second Confusion:* Next, LFSR updates the feedback coefficients with direct responses $G_2 = (r_1^*, r_2^*, r_3^*) = (101)$. Consequently, $C_4^* = S^4 = (s_0^4, s_1^4, s_2^4, s_3^4)$, where $s_0^4 = s_3^3 = 1$, $s_1^4 = s_0^3 \oplus s_3^3 = 0$, $s_2^4 = \neg s_1^0 = 0$, $s_3^4 = s_2^0 = 1$. In this case, we take $(r_4^*, r_5^*, r_6^*, r_7^*)$ as the 4-bit final response R.

Attackers can only eavesdrop on the original challenge $C = (0110)$ and the 4-bit final response $R = (0111)$. Therefore, owing to the 1-bit response in the first confusion poisoning the CRP set, the direct correlation between C and R has been cut successfully, enhancing the prediction difficulty of attackers significantly.

V. Performance Evaluations and Security Analysis

This section presents the performance evaluations of the proposed FLAM-PUF on 64-stage and 128-stage variants via Python simulation. The simulation and modeling attacks are implemented using Python 3.9 and executed on Windows 10 with 16 GB of main memory, an Intel Core i7-7700HQ CPU (2.8 GHz), and a Samsung 981A SSD of 512G. We used the Python-based APUF simulation model built in [48] and [54]. In this model, each stage delay parameter of APUF follows Gaussian random distribution $(\mu = 0.1, \delta = 1)$, meanwhile we have considered additive random noise with $N(0, 0.01)$ [14], [55].

A. Uniformity

Uniformity describes the ratio of 1 to 0 in the response. Actually, if the "1/0" ratio is 50%, the uniformity of a PUF is ideal. The uniformity can be measured by the hamming weight (HW) of the response sequence, which is defined as follows:

$$\text{HW}(R) = \sum_{i=1}^{N} r_i \tag{15}$$

where r_i represents the ith bit in N-bit response R. The probability of 1 in K responses with N-bits is

$$P_1 = \frac{1}{K} \sum_{j=1}^{K} \sum_{i=1}^{N} \frac{\text{HW}(R_j)}{N} \times 100\%. \tag{16}$$

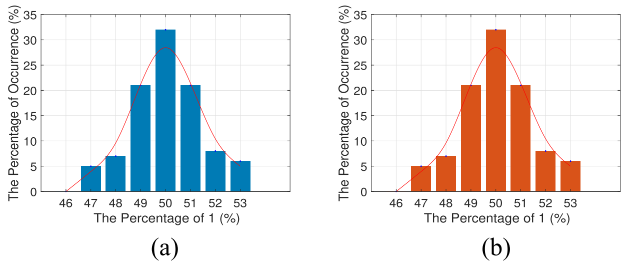

Fig. 7. Uniformity of (a) 64-stage and (b) 128-stage FLAM-PUF.

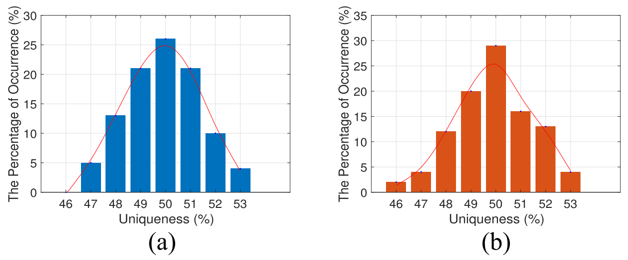

Fig. 8. Uniqueness of (a) 64-stage and (b) 128-stage FLAM-PUF.

Fig. 7(a) and (b) illustrates the distribution of P_1 of 100 64-stage and 100 128-stage FLAM-PUFs with $K = 1000$ for each entity, respectively. The average value of P_1 is about 49.73% and 49.99% for 64-stage and 128-stage FLAM-PUFs, respectively. As we can see, the distribution of P_1 of 64-stage FLAM-PUFs falls within the range of 45%–53%, and over 30% of P_1 values roughly equal the ideal value 50%. For 128-stage FLAM-PUFs, all P_1 fall within 46%–54%, and over 70% of them are within the range of 49%–51%.

B. Uniqueness

Even with the same structure and challenges, different PUF entities will generate different responses. Uniqueness indicates the difference between different PUFs, which is defined with the inter Hamming distance (HD)

$$\mathrm{HD}(R_i, R_j) = \sum_{n=1}^{N} \big(R_i[n] \oplus R_j[n]\big) \tag{17}$$

where R_i and R_j indicate the responses generated by two different PUF entities, PUF_i and PUF_j, under the same challenges. N represents the total number of bits in response. For K PUFs with N-bit response, the normalized inter HD is commonly employed to measure the uniqueness in

$$\mathrm{HD}_{\mathrm{nor}} = \frac{2}{K(K-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\mathrm{HD}(R_i, R_j)}{N} \times 100\%. \tag{18}$$

The ideal uniqueness is 50% for all PUF entities, which means half of the bits between two responses are different. Let $K = 2$, Fig. 8(a) and (b) plots the distribution of $\mathrm{HD}_{\mathrm{nor}}$ for 100 pairs of 64-stage and 128-stage FLAM-PUFs with 1000 challenges for each pair, respectively. The average value of $\mathrm{HD}_{\mathrm{nor}}$ is 49.81% and 49.85% for 64-stage and 128-stage FLAM-PUFs, respectively. All $\mathrm{HD}_{\mathrm{nor}}$ of 64-stage FLAM-PUFs fall within 47%–53%, and over 26% of them equal the ideal value 50%. For 128-stage FLAM-PUFs, all $\mathrm{HD}_{\mathrm{nor}}$ fall within 46%–53%, and more than 62% of them are within 49%–51%.

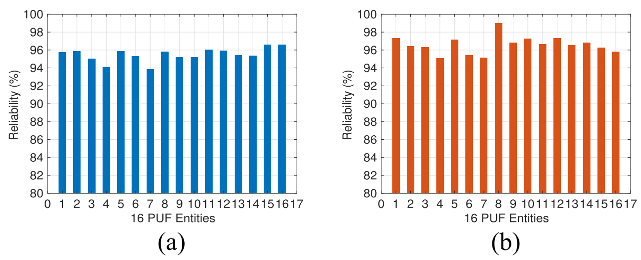

Fig. 9. Reliability of (a) 64-stage and (b) 128-stage FLAM-PUF.

C. Reliability

A PUF is expected to produce the same responses when receiving the same challenges under different environmental conditions. The reliability is calculated as follows:

$$\mathrm{Reliability} = 1 - \frac{1}{S} \sum_{i=1}^{S} \frac{\mathrm{HD}(R_i, R_{i,p})}{N} \times 100\%. \tag{19}$$

A PUF obtains S responses with n bit from the same challenge. $HD(R_i, R_{i,p})$ represents the HD between the sampled value and the standard value. Ideally, the reliability of a PUF is expected to be 100%. Fig. 9(a) and (b) measures the average reliability of 16 64-stage and 128-stage FLAM-PUFs under the same 100 challenges for 100 runs, respectively. All of the reliabilities of PUF entities are higher than 94%. For example, the average reliability for 64-stage and 128-stage FLAM-PUFs is 95.59% and 96.58%, respectively.

D. Threat Model and Security Analysis

Threat Model: As shown in Fig. 2, we first present the adversary model that is adopted in this work and then analyze the security of our approach in the presence of adversaries.

1) We assume that attackers can obtain some crucial initial information, including the circuit structure of FLAM-PUF, the position of the feedback point, and the initial state G_1 of the first confusion. In the IC supply chain, the foundry, the test facility, and the end user may all be untrusted attackers. Hence, attackers may directly access or use reverse engineering to master the circuit net list and circuit functions (including feedback point locations and initial information).
2) Attackers may also be capable of getting part of the original CRPs through an untrusted channel. For example, in PUF-based authentication, a verifier sends a challenge to a prover. The prover calculates the corresponding response by the PUF entity and sends it back to the verifier. An attacker can obtain this CRP if the verifier and prover transmit the challenge and response in plain text on an untrusted channel [56]. The direct CRP of APUF is essentially the intermediate result of calculation, which is only transmitted and temporarily stored inside the chip and will not reach the outside channel through the interface. In consequence, attackers cannot directly access the FLAM-PUF's internal circuit,

especially the original CRPs of APUF, e.g., the critical 1-bit response feedback.

3) However, the state of LFSR in FLAM-PUF is not allowed to be all 0 [21].

Security Analysis: We take the Galois LFSR as an example to analyze the security of FLAM-PUF.

1) The reconfigurable LFSR expands the original challenge C to a group of direct challenges. Usually, static LFSR is exploited to generate sequences as new challenges with a fixed algorithm. However, if polynomial coefficients $G(x)$ are not set properly, the cycles for LFSR to generate a repeatable sequence may be shortened, which will significantly weaken the obfuscation and make modeling attacks easier to be successfully launched. In contrast, LFSR can generate multiple CRPs for the APUF, i.e., $\{C_0^*, r_0^*\}, \{C_1^*, r_1^*\}, \{C_2^*, r_2^*\}, \ldots$ Since APUF is subject to ML-based modeling attacks, an attacker can obtain the direct CRPs from the APUF. Nevertheless, in our scheme, the attacker can only access the original challenge C of the first CRP $\{C_0^*, r_0^*\}$ and the final response r of the last CRP, not the direct challenges and responses of APUF. As a result, the attacker cannot accurately model the FLAM-PUF based on the misleading CRP mapping correlations, owing to a vast input space of C^* and the indirect output r.

2) The final response r or R is a strongly nonlinear function of the original challenge C. The final response can be calculated by the XORed several products of the time delay feature vector $\vec{w}$ and the feedback coefficient G_1, G_2. An algorithm such as LR [14], which relies on a linear function to fit the mapping between the original challenge and the final response, cannot achieve the intended prediction accuracy in our scheme, which has been clearly shown in Section V-E. Notably, we feed the response back to LFSR to construct a loop circuit structure, which can simultaneously achieve strong space and timing confusion. Time confusion is one of FLAM-PUF's key characteristics, which benefits from a loop process, causing the prediction of a r to depend on all the preceding responses. Thus, the double confusion of space and time can boost the system complexity and thus enhance the resistance to modeling attacks.

3) The reconfigurable LFSR randomly updated by 1-bit response feedback can implant problematic correlations into the target model of attackers. We feed back the 1-bit direct response to randomly update one feedback coefficient to introduce (1/2) uncertainty in each cycle. Specifically, the circuit structure and the position of the feedback point are public, only if an attacker does not know the correct r_i^*. For example, as shown in Fig. 4, there are two possible CRPs in the next cycle, which will introduce some poisoned data to the dataset for the ML-based modeling attackers.

Consequently, the response feedback brings at least an exponential uncertainty to the system. For an n-bit response, given that the total number of poisoned CRPs is n, the probability of an attacker guessing all n bit responses should be $(1/2)^n$. For instance, the probability is $5.42e$-20 and $2.93e$-39 when $n = 64$ and 128, respectively. On the other hand, suppose that the attacker does not know the location of the feedback point, there will be $(n-1)$ possible feedback coefficients. The attack difficulty will approximately increase by n times.

E. Resistance to Modeling Attacks

In this section, we evaluate the prediction accuracy of modeling attacks under four well-known ML algorithms, i.e., LR, SVM, CMA-ES, and DNN, to verify the resistance of FLAM-PUF. Our experiments have modeled three scales of FLAM-PUF, i.e., 32-stage, 64-stage, and 128-stage, generating the 1-bit final response at the $(n + 1)$th and $2n$th cycle, respectively. In addition, the experiment collects 10^6 CRPs from each FLAM-PUF instance for testing.

The SVM attack is built on Python's sklearn library function *sklearn.svm.SVC*. All parameters are set by default, including the regularization parameter $C = 0.1$ and the kernel type $kernel$ = "rbf." We employ ten-fold cross-validation to test the accuracy of the algorithm. The LR attack takes advantage of Python's sklearn library function *sklearn.linear_model.LogisticRegression*. Similarly, all parameters are set by default, where $C = 0.1$ and $kernel$ = rbf. It utilizes the ten-fold cross-validation to test the accuracy of the algorithm as well. The CMA-ES attack exploits Python's cma library function *cma.CMAEvolutionStrategy*. Most parameters are set by default. All initial solutions are set to 0, and the initial standard deviation $sigma0 = 0.5$. Moreover, we take 70% of the CRP set as the training set and the remaining 30% as the test set. For the DNN attack, we construct a four-layer neural network, also taking 70% of the CRP set as the training set and the remaining 30% as the test set.

All initial solutions are set to 0, and the initial standard deviation $sigma0 = 0.5$. Moreover, we take 70% of the CRP set as the training set and the remaining 30% as the test set. For the DNN attack, we construct a four-layer neural network, also taking 70% of the CRP set as the training set and the remaining 30% as the test set.

Fig. 10 demonstrates that in the presence of the modeling attacks with 10^6 CRPs data set, the prediction accuracy values of the FLAM-PUF fluctuate around 50%, and they all fall below 55% without an apparent rising trend, indicating its excellent resistance to state-of-the-art modeling attacks. It can be seen from the four figures that when the data set is small ($\leq 10^4$), the prediction accuracy fluctuates relatively remarkably, owing to the uneven data distribution. The fluctuation of the prediction accuracy tends to converge around 50% as the number of CRPs exceeds 10^5. It is also noteworthy that it is sufficient to use up to 10^6 CRPs to train and verify the security of our solution.

Specifically, the prediction accuracy of the LR algorithm is not less than 50% and its fluctuation range is the smallest. To solve the problem that the LR and SVM may fall into the local optimal solution, CMA-ES updates the parameters in a randomized manner, but its convergence is not as good as LR and SVM. The model established by DNN in a black-box manner may significantly mismatch with the actual CRPs mapping, and most prediction results are lower than 50%.

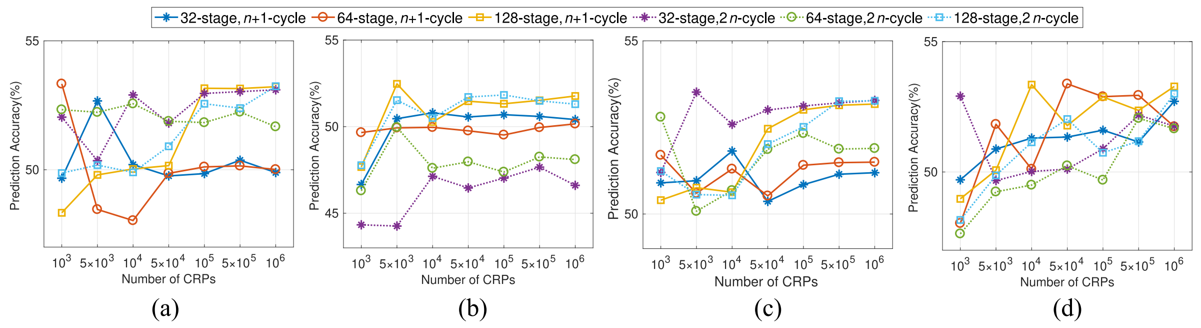

Fig. 10. Resistance of FLAM-PUF under four ML algorithms (i.e., LR, SVM, CMA-ES, and DNN). (a) Resistance against SVM attacks. (b) Resistance against LR attacks. (c) Resistance against CMA-ES attacks. (d) Resistance against DNN attacks.

TABLE II
COMPARISON ON PRIMARY METRICS BETWEEN FLAM-PUF AND PRIOR STRONG PUFS

stage number	PUF type	uniformity	uniqueness	reliability	prediction accuracy			
					SVM	LR	CMA-ES	DNN
128	**FLAM-PUF**	49.99%	49.85%	96.58%	48.1%-53.3%	50.6%-52.8%	48.3%-53.2%	46.3%-50.2%
64	**FLAM-PUF**	49.73%	49.81%	95.59%	47.6%-53.4%	50.4%-53.4%	50.0%-53.3%	47.6%-52.4%
128	CT PUF [17]	49%-50%	48%-50%	-	-	-	50%-60%	-
64	CT PUF [17]	49%-50%	49%-50%	$\geq$ 92%	-	-	50%-57%	-
128	[57]	49.8%	-	99.74%*	$\approx$ 50%	$\approx$ 50%	$\approx$ 50%	49%-51%
64	SR-PUF [22]	49.96%-50.07%	49.98%-50.01%	91.87%-96.12%*	-	49.91%	50.20%	50.28%

Thanks to the response feedback-controlled LFSR, the FLAM-PUF input-output mapping is a robust nonlinear function, resisting the linear modeling attacks for feature vectors. The LR and SVM algorithms are more suitable for linear relations, with a poor fitting effect on nonlinear functions. The CMA-ES algorithm can solve nonlinear nonconvex functions, yet the convergence speed is slow due to the random update strategy. Although DNNs can build nonlinear models as an empirical black-box model at the cost of increasing parameters by manual setting, it lacks targeted analysis and thus becomes ineffective if used for modeling our proposed PUFs.

To summarize, as shown in Table II, the uniqueness and uniformity of our FLAM-PUF are all close to the ideal value of 50%. The reliability is higher than 95%, which is comparable to other works. The prediction accuracy of the four typical attack algorithms is lower than 55%, which is approximate to random guessing. Our proposed PUF can well balance safety and performance. It is worth noting that the PUFs proposed in [22] and [57] both adopt a reliability enhancement method.

F. Hardware Cost Comparisons

The CMOS technology of 65 nm has become the *de-facto* standard in mainstream processor manufacturers these years. According to [21], Table III lists the gate equivalent (GE) in 65-nm CMOS technology. A GE stands for a unit of measure which allows specifying manufacturing-technology-independent complexity of digital electronic circuits. In Table IV, we compare the number of GEs of FLAM-PUF with state-of-the-art robust strong PUFs under the same attack prediction accuracy (i.e., roughly 50%).

First, we take 128-stage APUF as an example to illustrate the calculation of GE. In the 128-stage APUF, the multiplexers chain consists of $128 \times 2 = 256$ MUXs, and the arbiter is

TABLE III
GE HARDWARE OVERHEAD IN 65-NM CMOS TECHNOLOGY

Gate	2-input NAND	2-input AND	2-input XOR	2-to-1 MUX	flip-flop
GE	1	1.5	2.5	2.5	6.25

realized with a flip-flop. According to Table III, it is easy to calculate the total number of GEs required for a 128-stage APUF is equal to $(256 \times 2.5 + 1 \times 6.25) \approx 646$. Besides APUF, the LFSR incurs the primary hardware cost of FLAM-PUF, which is roughly $3\times$ the number of GEs of a standard APUF. For instance, a 128-stage reconfigurable LFSR in our design includes 128 flip-flops as registers, 127 2-input XOR, 126 2-input AND, and 2 2-input NAND as feedback module. As Table IV shows, the total number of GEs required for a 128-stage FLAM-PUF is $(256 \times 2.5 + 129 \times 6.25 + 127 \times 2.5 + 126 \times 1.5 + 2 \times 1) \approx 1955$. In contrast, x-XOR APUF is resistant to LR attacks only when x is very large, but the hardware overhead overgrows as x increases. The 128-stage x-XOR APUF with $x \leq 7$ can be successfully attacked by LR algorithm [57]. While the 8-XOR APUF calculating the responses of 8 APUFs via XOR operations will incur about $8\times$ the number of GEs of a single APUF, i.e., about 5100 GEs. In Ising-PUF [15], each cell PUF includes a 4-stage APUF and corresponding control circuits. The 128-stage Ising-PUF costs about 7000 GEs, i.e., $3.6\times$ the number of GEs of our scheme.

Similarly, 64-stage 8-XOR APUF costs about half of the number of GES of 128-stage 8-XOR APUF. The (x, y)-iPUF combines an n stage x-XOR APUF with an $(n+1)$-stage y-XOR. The hardware overhead of a (x, y)-iPUF is close to $(x + y)\times$ of an APUF. The (1, 9)-iPUF of 64-stage and (8, 8)-iPUF of 64-stage in [54] require about $10\times$ and $16\times$ the number of GEs of a 64-stage APUF, respectively. As we can see in

TABLE IV
HARDWARE COST COMPARISON OF FLAM-PUF WITH OTHER ROBUST PUFS UNDER THE SAME ATTACK PREDICTION ACCURACY

stage number	PUF type	the number of gates						GE
		2-input NAND	2-input AND	2-input XOR	2-to-1 MUX	flip-flop	other gates	
128	**FLAM-PUF**	2	126	127	256	129	-	**1955**
128	8-XOR APUF [24]	-	-	-	2048	8	1	$\approx$ 5100
128	Ising-PUF [15]	128	384	-	1408	384	$\approx$ 384	$\approx$ 7000
64	**FLAM-PUF**	2	62	63	128	65	-	**979**
64	8-XOR APUF [24]	-	-	-	1024	8	1	$\approx$ 2600
64	Ising-PUF [15]	64	192	-	704	192	$\approx$ 192	$\approx$ 3500
64	(1,9)-iPUF [54]	-	-	-	1298	10	1	$\approx$ 3300
64	(8,8)-iPUF [54]	-	-	-	2064	16	2	$\approx$ 5300
64	CT PUF [17]	-	-	-	-	486	731 LUT	$\geq$ 10, 000

Table IV, the Ising-PUF, (1, 9)-iPUF, and (8, 8)-iPUF of 64-stage need about 3.6×, 3.3×, and 5.3× of hardware cost of the 64-stage FLAM-PUF, respectively. The experimental results in [17] show that a 64-stage CT-PUF needs 731 look-up tables (LUTs) and 486 flip-flops. Each LUT is roughly equivalent to 10–20 prime logical gates, and the number of GEs required for the entire circuit is larger than 10 000.

In summary, compared with classic robust strong PUFs, i.e., x-XOR APUF, Ising-PUF, iPUF, and CT PUF, FLAM-PUF can reduce at least 62% hardware overhead in 64-stage and 128-stage.

Evaluation Summary: FLAM-PUF can achieve excellent resistance to widely used modeling algorithms by introducing nonlinear obfuscation and exponential uncertainty. When the attack model is very close to the PUF mapping, the prediction accuracy of the attack algorithm on the test set is high, indicating a successful attack. In the experiment, the results for the 1-bit response can only be 1 or 0. Hence, when the prediction accuracy is close to 50%, indicating a failing attack. FLAM-PUF as a defense entity makes the prediction accuracy of all four attack algorithms around 50%, showing that it can well resist existing modeling attacks. FLAM-PUF mainly consists of a reconfigurable Galois LFSR and an APUF. Compared with state-of-the-art robust strong PUFs, FLAM-PUF is lightweight and can achieve more than a 62% reduction in hardware overhead in 64-stage and 128-stage.

VI. CONCLUSION

We presented FLAM-PUF, the first response feedback-based lightweight PUF with high anti-ML-attack ability. FLAM-PUF was mainly composed of a reconfigurable LFSR and an APUF. Specifically, FLAM-PUF leverages a cost-effective feedback loop structure to control and update the LFSR configuration dynamically, achieving at least a 62% reduction in hardware cost compared with state-of-the-art robust strong PUFs. In addition to exploiting a n-bit response feedback-controlled reconfigurable Galois LFSR to enlarge the original challenge space of the APUF, FLAM-PUF feeds back a 1-bit response in every cycle to intentionally poison the data of the CRP set for training, by randomly updating one coefficient of feedback polynomial to implant problematic correlations into the target model of attackers. The extensive experimental results demonstrated that FLAM-PUF achieves near-optimal uniformity, uniqueness, and reliability, and works well under standard attack models with public crucial initial information. The prediction accuracy of modeling attacks against FLAM-PUF is nearly 50% under the four widely used ML algorithms, i.e., LR, SVM, DNN, and CMA-ES. On the other hand, our solution requires at least $(n + 1)$ cycles to generate the final responses. We next intend to design more efficient PUF solutions in both area and time delay.

REFERENCES

[1] H. P. Breivold and K. Sandström, "Internet of Things for industrial automation—Challenges and technical solutions," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, 2015, pp. 532–539.

[2] T. Malche and P. Maheshwary, "Internet of Things (IoT) for building smart home system," in *Proc. Int. Conf. IoT Soc. Mobile Anal. Cloud (I-SMAC)*, 2017, pp. 65–70.

[3] D. Vishal, H. S. Afaque, H. Bhardawaj, and T. K. Ramesh, "IoT-driven road safety system," in *Proc. Int. Conf. Elect. Electron. Commun. Comput. Optim. Techn. (ICEECCOT)*, 2017, pp. 1–5.

[4] M. Neyja, S. Mumtaz, K. M. S. Huq, S. A. Busari, J. Rodriguez, and Z. Zhou, "An IoT-based E-health monitoring system using ECG signal," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.

[5] "IoT security—A guide to a safer connected experience." 2021. [Online]. Available: https://www.appsealing.com/iot-security/

[6] E. Dubrova, M. Näslund, G. Selander, and F. Lindqvist, "Message authentication based on cryptographically secure CRC without polynomial irreducibility test," *Cryptography Commun.*, vol. 10, no. 2, pp. 383–399, 2018.

[7] W. Trappe, R. Howard, and R. S. Moore, "Low-energy security: Limits and opportunities in the Internet of Things," *IEEE Security Privacy*, vol. 13, no. 1, pp. 14–21, Jan./Feb. 2015.

[8] D. Pishva, "IoT: Their conveniences, security challenges and possible solutions," *Adv. Sci. Technol. Eng. Syst. J*, vol. 2, no. 3, pp. 1211–1217, 2017.

[9] M. Šimon, L. Huraj, and T. Horák, "DDoS reflection attack based on IoT: A case study," in *Cybern. Algorithms Intell. Syst.*, R. Silhavy, Ed. Cham, Switzerland: Springer Int., 2019, pp. 44–52.

[10] Y. Hu *et al.*, "STT-MRAM-based reliable weak PUF," *IEEE Trans. Comput.*, vol. 71, no. 1, pp. 1564–1574, Jul. 2022.

[11] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong PUFs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2138–2151, Oct. 2020.

[12] S. Bhunia and M. Tehranipoor, "Chapter 12—Hardware security primitives," in *Hardware Security*, S. Bhunia and M. Tehranipoor, Eds. London, U.K.: Morgan Kaufmann, 2019, pp. 311–345. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128124772000174

[13] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[14] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.

[15] H. Awano and T. Sato, "Ising-PUF: A machine learning attack resistant PUF featuring lattice like arrangement of arbiter-PUFs," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2018, pp. 1447–1452.

[16] V. B. Suresh, R. Kumar, and S. Mathew, "A 0.% BER, machine-learning resistant 10^{28} challenge-response PUF in 14nm CMOS featuring stability-aware adversarial challenge selection," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–3.

[17] J. Zhang, C. Shen, Z. Guo, Q. Wu, and W. Chang, "CT PUF: Configurable tristate PUF against machine learning attacks for IoT security," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14452–14462, Aug. 2022.

[18] L. Yu, X. Wang, F. Rahman, and M. Tehranipoor, "Interconnect-based PUF with signature uniqueness enhancement," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 339–352, Feb. 2020.

[19] Y. Wang, C. Wang, C. Gu, Y. Cui, M. O'Neill, and W. Liu, "A dynamically configurable PUF and dynamic matching authentication protocol," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1091–1104, Apr.–Jun. 2022.

[20] M. Elshamy and H.-G. Stratigopoulos, "Neuron-PUF: Physical Unclonable function based on a single spiking neuron," in *Proc. 27th IEEE Int. Symp. On-Line Test. Robust Syst. Design*, 2021, pp. 1–6.

[21] E. Dubrova, O. Näslund, B. Degen, A. Gawell, and Y. Yu, "CRC-PUF: A machine learning attack resistant lightweight PUF construction," in *Proc. IEEE Eur. Symp. Security Privacy Workshops (EuroS & PW)*, 2019, pp. 264–271.

[22] S. Hou, D. Deng, Z. Wang, J. Shi, S. Li, and Y. Guo, "A dynamically configurable LFSR-based PUF design against machine learning attacks," *CCF Trans. High Perform. Comput.*, vol. 3, no. 1, pp. 31–56, 2021.

[23] U. Rührmair *et al.*, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013.

[24] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, 2007, pp. 9–14.

[25] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks," in *Proc. IACR Trans. Cryptograph. Hardw. Embedded Syst.*, 2019, pp. 243–290.

[26] Q. Wang, M. Gao, and G. Qu, "A machine learning attack resistant dual-mode PUF," in *Proc. Great Lakes Symp. VLSI*, 2018, pp. 177–182.

[27] X. Xu, U. Rührmair, D. E. Holcomb, and W. Burleson, "Security evaluation and enhancement of bistable ring PUFs," in *Proc. Int. Workshop Radio Freq. Identif. Security Privacy Issues*, 2015, pp. 3–16.

[28] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 889–902, Jun. 2015.

[29] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe, "PUF-FSM: A controlled strong PUF," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1104–1108, May 2018.

[30] Y. Gao, S. F. Al-Sarawi, D. Abbott, A.-R. Sadeghi, and D. C. Ranasinghe, "Modeling attack resilient reconfigurable latent obfuscation technique for PUF based lightweight authentication," 2017, *arXiv:1706.06232*.

[31] J. Ye, Y. Hu, and X. Li, "RPUF: Physical unclonable function with randomized challenge to resist modeling attack," in *Proc. IEEE Asian Hardw. Oriented Security Trust (AsianHOST)*, 2016, pp. 1–6.

[32] S.-J. Wang, Y.-S. Chen, and K. S.-M. Li, "Adversarial attack against modeling attack on PUFs," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[33] J. Zhang, L. Wan, Q. Wu, and G. Qu, "DMOS-PUF: Dynamic multi-key-selection obfuscation for strong PUFs against machine learning attacks," 2018, *arXiv:1806.02011*.

[34] D. Suzuki and K. Shimizu, "The glitch PUF: A new delay-PUF architecture exploiting glitch shapes," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)*, 2010, pp. 366–382.

[35] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2015, pp. 653–658.

[36] R. Kumar and W. Burleson, "On design of a highly secure PUF based on non-linear current mirrors," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST)*, 2014, pp. 38–43.

[37] Z. He, M. Wan, J. Deng, C. Bai, and K. Dai, "A reliable strong PUF based on switched-capacitor circuit," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 6, pp. 1073–1083, Jun. 2018.

[38] F. Afghah, B. Cambou, M. Abedini, and S. Zeadally, "A reram physically unclonable function (ReRAM PUF)-based approach to enhance authentication security in software defined wireless networks," *Int. J. Wireless Inf. Netw.*, vol. 25, no. 2, pp. 117–129, 2018.

[39] S. Khaleghi, P. Vinella, S. Banerjee, and W. Rao, "An stt-MRAM based strong PUF," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit. (NANOARCH)*, 2016, pp. 129–134.

[40] S. Jeloka, K. Yang, M. Orshansky, D. Sylvester, and D. Blaauw, "A sequence dependent challenge-response PUF using 28nm SRAM 6T bit cell," in *Proc. IEEE Symp. VLSI Circuits*, 2017, pp. 270–271.

[41] X. Xi, H. Zhuang, N. Sun, and M. Orshansky, "Strong subthreshold current array PUF with 2^{65} challenge-response pairs resilient to machine learning attacks in 130nm CMOS," in *Proc. Symp. VLSI Circuits*, 2017, pp. 268–269.

[42] J. Liu, Y. Zhu, C.-H. Chan, and R. P. Martins, "A 0.46pJ/bit ultralow-power entropy-preselection-based strong PUF with worst-case BER $\leq 6.7 \times 10^{-6}$," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, 2021, pp. 1–3.

[43] Y.-C. Lai, C.-Y. Yao, S.-H. Yang, Y.-W. Wu, and T.-T. Liu, "A robust area-efficient physically unclonable function with high machine learning attack resilience in 28-nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 1, pp. 347–355, Jan. 2022.

[44] U. Rührmair *et al.*, "Efficient power and timing side channels for physical unclonable functions," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2014, pp. 476–492.

[45] X. Xu and W. Burleson, "Hybrid side-channel/machine-learning attacks on PUFs: A new threat?" in *Proc. IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2014, pp. 1–6.

[46] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency Comput. Pract. Exp.*, vol. 16, no. 11, pp. 1077–1098, 2004.

[47] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure pufs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 670–673.

[48] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Security*, 2002, pp. 148–160.

[49] S. Saranyadevi and M. Thangavel, "A low power structure design of 2D-LFSR and encoding technique for bist," *Int. J. Adv. Sci. Technol.*, vol. 18, pp. 11–22, May 2010.

[50] C.-I. Chen and K. George, "Configurable two-dimensional linear feedback shifter registers for parallel and serial built-in self-test," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 4, pp. 1005–1014, Aug. 2004.

[51] S. Mishra, R. R. Tripathi, and D. K. Tripathi, "Implementation of configurable linear feedback shift register in VHDL," in *Proc. Int. Conf. Emerg. Trends Elect. Electron. Sustain. Energy Syst. (ICETEESES)*, 2016, pp. 342–346.

[52] N. Hansen, "The CMA evolution strategy: A tutorial," 2016, *arXiv:1604.00772*.

[53] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2015, pp. 535–555.

[54] N. Wisiol *et al.*, "Splitting the interpose PUF: A novel modeling attack strategy," in *Proc. IACR Trans. Cryptograph. Hardw. Embedded Syst.*, 2020, pp. 97–120.

[55] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST)*, 2013, pp. 137–142.

[56] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[57] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Proc. Int. Workshop Radio Freq. Identif. Security Privacy Issues*, 2015, pp. 17–31.