A Modeling Attack Resistant Deception Technique for Securing Lightweight-PUF-Based Authentication

Chongyan Gu, *Member, IEEE*, Chip-Hong Chang, *Fellow, IEEE*, Weiqiang Liu, *Senior Member, IEEE*, Shichao Yu, Yale Wang, and Máire O'Neill, *Senior Member, IEEE*

***Abstract*—Silicon physical unclonable function (PUF) has emerged as a promising spoof-proof solution for low-cost device authentication. Due to practical constraints in preventing phishing through a public network or insecure communication channels, simple PUF-based authentication protocol with unrestricted queries and transparent responses is vulnerable to modeling and replay attacks. In this article, we present a modeling attack resistant PUF-based mutual authentication scheme to mitigate the practical limitations in applications where a resource-rich server authenticates a device with no strong restriction imposed on the type of PUF design or any additional protection on the binary channel used for the authentication. Our scheme uses an active deception protocol to prevent machine learning (ML) attacks on a device with a monolithic integration of a genuine strong PUF (SPUF), a fake PUF, a pseudorandom number generator (PRNG), a register, a binary counter, a comparator, and a simple controller. The hardware encapsulation makes the collection of challenge–response pairs (CRPs) easy for model building during enrollment but prohibitively time consuming upon device deployment through the same interface. A genuine server can perform a mutual authentication with the device using a combined fresh challenge contributed by both the server and the device. The message exchanged in clear cannot be manipulated by the adversary to derive unused authentic CRPs. The adversary will have to either wait for an impractically long time to collect enough real CRPs by directly querying the device or the ML model derived from the collected CRPs will be poisoned. The false PUF multiplexing is fortified against the prediction of waiting time by doubling the time penalty for every unsuccessful guess. Our implementation results on field-programmable gate array (FPGA) device and security analysis have corroborated the low hardware overheads and attack resistance of the proposed deception protocol.**

***Index Terms*—Authentication protocol, deception protocol, machine learning attacks, physical unclonable function (PUF), Poison attack.**

I. Introduction

WHILE Internet of Things (IoT) revolutionizes our lives through remote healthcare, autonomous vehicles, and smart homes, it also brings security issues. The large number of devices open up new attack vectors, as exemplified by the IoT-based distributed denial-of-service (DDoS) attack on Dyn, that brought down Twitter, SoundCloud, Spotify, Reddit, and a host of other sites [1]. Providing security to IoT devices is a major challenge as conventional security approaches, based on provably secure cryptographic algorithms are too resource intensive for implementation on these devices.

A physical unclonable function (PUF), is a security primitive that utilizes intrinsic manufacturing process variations to generate a unique digital fingerprint. A comprehensive review of PUF can be found in [2]. As this natural variation among silicon dies is outside the control of the manufacturer, PUFs are inherently difficult to clone, and possess additional tamper-evident properties [3], [4]. PUFs can produce unique keys on-the-fly, which reduces the risk of physical attacks and saves hardware resources. These properties open up interesting opportunities for higher level security protocols, such as key generation and device authentication for both application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA)-based devices.

The initial proposal of using a Hamming distance (HD) threshold τ comparison for lightweight PUF-based authentication protocol was proposed in [5]. Since then, similar PUF-based authentication protocols have been derived to endow linearly sized strong PUF (SPUF) circuits with an exponentially large challenge–response pair (CRP) capacity for device authentication. Unfortunately, SPUFs used for device authentication with limited nonlinear mixing have been shown to be vulnerable to machine learning (ML)-based modeling attacks. Masquerade attacks can be perpetrated through malicious nodes and unprotected communication links of ad hoc networks to efficiently collect a large number of CRPs to model a PUF. To prevent modeling attacks, many countermeasures [6]–[8] have been proposed, e.g., increasing the circuit complexity of PUF. These approaches tend to degrade other quality such as reliability of the underlying PUF. They are not resilient against replay and man-in-the-middle (MITM) attacks without additional protection to track and prevent the

Manuscript received March 29, 2020; revised July 1, 2020 and October 6, 2020; accepted October 24, 2020. Date of publication November 10, 2020; date of current version May 20, 2021. This work was supported in part by the Singapore MOE Tier 1 under Grant MOE2018-T1-001-131 RG87/18(S); in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/N508664/-CSIT2; and in part by the National Natural Science Foundation of China under Grant 62022041 and Grant 61771239. This article was recommended by Associate Editor W. Hu. *(Corresponding authors: Chongyan Gu; Weiqiang Liu.)*

Chongyan Gu, Shichao Yu, and Máire O'Neill are with the Centre for Secure Information Technologies, Institute of Electronics, Communications, and Information Technology, Queen's University Belfast, Belfast BT3 9DT, U.K. (e-mail: cgu01@qub.ac.uk; syu08@qub.ac.uk; m.oneill@ecit.qub.ac.uk).

Chip-Hong Chang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: echchang@ntu.edu.sg).

Weiqiang Liu and Yale Wang are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: liuweiqiang@nuaa.edu.cn; yalewang@nuaa.edu.cn).

Digital Object Identifier 10.1109/TCAD.2020.3036807

used CRPs from being reused. To defend against brute-force query by using existing SPUF for authentication of lightweight devices, e.g., IoT and wearable devices, the lockdown protocol [9] restricts the number of authentication events to limit the number of CRPs from being learned for modeling attack.

Unlike prior works, our proposed PUF-based authentication protocol does not simplistically rely on expensive error correction code (ECC) or crypto-algorithm on the device side to passively increase its attack resistance. Instead, it adopts an offensive defense strategy to cautiously retaliate and frustrate the adversaries by poisoning their training data. This is achieved through a combination of hardware encapsulation and protocol-level approach to deceive or excessively delay the adversary attempting to collect the CRPs by brute-force queries made on the device enrollment protocol interface. The proposed deception protocol enables a resource-rich server to authenticate a device with no strong restriction imposed on the type of PUF design or any additional protection on the binary channel used for the authentication. Specifically, re-enrollment for model calibration is permitted by our model-based authentication. It can be made more secure against direct CRP collection through the enrollment protocol for model building attack.

A hardware implementation for the proposed deception protocol is demonstrated on a Xilinx Artix-7 FPGA to validate its hardware efficiency. Using the most recent published experimental platforms[10], its resistance against modeling attacks is evaluated by linear regression (LR) and covariance matrix adaptation evolution strategies (CMA-ES). The effectiveness of the fake PUF in reducing the successful prediction rate is also evaluated.

The remainder of this article is organized as follows. Section II reviews the vulnerability of SPUFs to modeling attacks and identifies the gaps in existing PUF-based authentication schemes. Section III provides the preliminaries of PUF-based authentication and modeling attacks. The proposed deception protocol is presented in Section IV. It is evaluated against both linear regression (LR) and covariance matrix adaptation evolution strategies (CMA-ES) attacks in Section V. An FPGA-based implementation of the proposed deception protocol is presented in Section VI. A comparison between the proposed deception protocol and other lightweight authentication protocols is presented in Section VII. Finally, conclusions are drawn in Section VIII.

II. Attacks on SPUFs

PUF architectures are broadly categorized into weak and strong in [11] based solely on their number of unique CRPs per bit cell. Weak PUFs have a limited CRP space. They are more suited as a random key generator or for seeding a pseudorandom number generator (PRNG), where the response never leaves the chip and is only accessed as required. In contrast, SPUFs have a large number of possible CRPs. The response bitstream returned from a random sequence of challenges is unique to each challenge sequence and the physical PUF device. By design, this implies the requirement for a much larger entropy pool such that related challenges should not lead to related responses on the same device. Hence, SPUFs are preferred to weak PUFs for device authentication.

Most SPUF architectures based on linear additive functions have been shown to be vulnerable to ML attacks [12]–[15]. Therefore, the focus on SPUF research nowadays has been directed toward preventing ML attacks. At circuit level, the modeling complexity of the SPUF design has been increased by, e.g., XOR arbiter PUF (APUF) [6], feedforward APUF [7], lightweight secure APUF [8], and multi-PUF [16]. Most of these approaches achieve increased resistance to ML attacks by using more complex PUF architectures, e.g., [6] and [8], while preserving a desirably huge number of CRPs. Even then, it has been shown more recently that the workload of ML attacks can be reduced and the success rate can be raised through exploiting the weakness in their protocols [12]–[15], [17]. Side-channel information has also been demonstrated to aid PUF attack. PUF noise can be filtered to improve the signal-to-noise ratio for efficient side-channel attacks [18], [19]. A combination of side-channel analysis (SCA) and modeling attacks, exploiting the noise of PUFs, have been proposed to effectively crack the secret response of PUF [20], [21].

Delvaux *et al.* [22] presented a survey on entity authentication protocols using PUFs. It shows that most PUF protocols [23]–[27] are heavyweight or require complicated protocol operations that add to the hardware implementation area, power, and performance overheads. Two protocols, the slender PUF [28] and noise bifurcation [29], require neither an ECC nor a strong cryptographic algorithm. A true random number generator (TRNG) has been used instead to provide heuristic security against modeling attacks, which is difficult to validate. The idea of slowing down the read out of PUF response was first raised in [30]. The SHIC PUF proposed in [30] was realized by an emerging high density $10^5 \times 10^5$ crossbar memory with an intrinsically slow access speed of 100 b/s. It is assumed that the amount of independent structural information of SHIC PUF cannot be fully modeled without a complete circuit characterization. Hence, it takes $\approx$ 3 years to collect the full set of CRPs with this access speed. However, if only 10^5 instead of the complete set of CRPs is required to learn a PUF by ML techniques, they can be acquired in less than 20 min. As SHIC PUF is implemented by emerging semiconductor technology, it does not integrate well with conventional CMOS designs [9]. The incompatibility also raises constraints on the "nano–micro" link [30]. Moreover, there is a constraint on the intrinsic slowdown of SHIC PUF as it will impact the number of CRPs that can be practically enrolled. As SHIC is a weak PUF by the independent cell structure criterion, its CRP space grows only linearly as opposed to exponentially with array size. The cost of producing only $10^{10} \approx 2^{33}$ CRPs by SHIC PUF is significantly higher than a 128-stage APUF that can generate 2^{128} CRPs. Recently, Yu *et al.* [9] proposed two lockdown techniques to limit the number of authentication requests. The first lockdown protocol allows only unilateral authentication while the second requires a TRNG on the device side to generate a device nonce, c_D. As the responses r_1 and r_2 in the second lockdown protocol are sent in clear, the adversary can query the device to obtain a new device-side challenge c'_D and then adjust the server-side challenge c'_S until the linear-feedback shift register (LFSR) output $<c>$ is the same as one of the previously eavesdropped tuples (c_D, c_S, r_1). The adversary can then reply with $c'_S \| r_1$ to gain successful authentication. The viability of this is limited by the previously eavesdropped tuples and LFSR seed length. The tradeoff of the lockdown protocol is that it can only support a limited number of authentication requests for the verifier. The protocol relies on the hardness of XOR PUF in the field to support model-based authentication. However, its hardware cost increases and reliability reduces commensurately with an increasing number of XORs for higher ML resistance. Gao *et al.* [31] used a reconfigurable latent obfuscation technique to conceal and distort the relationship between CRPs. The pattern vectors for challenge and response obfuscation are selected by a random number generator (RNG), and are made latent and reconfigurable per authentication session. Nevertheless, the most recent report shows that both methods [9], [31] are vulnerable to the protocol attack [32]. A PUF-based mutual authentication protocol called PHEMAP has been proposed recently [33]. However, it has been shown to be vulnerable to impersonate, desynchronization, and traceability attacks [34]. Two multiplexer-based PUF (MPUF) variants (rMPUF and cMPUF) [35] were also proposed to resist reliability-based and cryptanalysis modeling attacks, respectively. However, they have been successfully attacked by approximation attack in [36] using an approximation algorithm-based artificial neuron network. This advanced attack falls under the machine learning-based approaches, which require the collection of a sufficient number of valid CRPs for training.

TABLE I
LIST OF FREQUENTLY USED NOTATIONS

Notation	Description
ID_i	Identifier of device i
n	Length of PUF challenge string
m	Length of PUF response string
$\boldsymbol{c}$	An n-bit challenge
$\langle c \rangle$	List of consecutive n-bit sub-challenges derived from c
$\boldsymbol{r}$	Enrolled m-bit response
$\tilde{\boldsymbol{r}}$	Reproduced m-bit response
(c, r)	A challenge-response pair with the response bits of r derived from $\langle c \rangle$
d	Number of CRPs stored in the server
$\mathsf{FHD}(a, b)$	Fractional Hamming distance between two binary strings a and b of the same length
τ	Fractional Hamming distance threshold
N_{CRP}	Number of CRPs needed for training a SPUF model
$\mathsf{SPUF^G}$	Genuine strong PUF enrolled in the server
$\mathsf{SPUF^F}$	Fake strong PUF not enrolled in the server
$\mathsf{PRNG}(a)$	Pseudo random number generator with seed a
$\mathsf{TRNG}(a)$	True random number generator that outputs a random bits
$a \,\|\, b$	Concatenation of binary strings a and b
$a \oplus b$	Bitwise XOR of binary strings a and b of the same length
T_w	Waiting time
$\mathsf{T_{os}}$	Dynamic component of T_w and $\mathsf{P_{os}}$ is its binary form
$\mathsf{T_{min}}$	Static component of T_w and $\mathsf{P_{min}}$ is its binary form
σ_{noise}	Standard deviation of environmental noise
Q	Counter's output
Z	Comparator's output
S	$\mathsf{SPUF^G}$ or $\mathsf{SPUF^F}$ selection signal

Device i		Server
	Enrollment $(1\times)$	
$OTP: \mathsf{ID} \leftarrow \mathsf{ID}_i$	$\xleftarrow{\mathsf{ID}_i}$	
$\mathsf{r}_{ij} \leftarrow \mathsf{SPUF}(\mathsf{c}_{ij})$	$\longleftrightarrow$	$(\mathsf{c}_{ij}, \mathsf{r}_{ij})$ with $\mathsf{c}_{ij} \leftarrow \mathsf{RNG}()$ and $j \in [1\ d]$
		$d_i \leftarrow d$
		$(\mathsf{c}, \mathsf{r}) \leftarrow (\mathsf{c}_{ij}, \mathsf{r}_{ij})$ with $j \leftarrow d_i$
		$d_i \leftarrow d_i - 1$
	Authentication $(d\times)$	
	$\xleftarrow{\mathsf{c}}$	
$\tilde{\mathsf{r}} \leftarrow \mathsf{SPUF}(\mathsf{c})$	$\xrightarrow{\tilde{\mathsf{r}}}$	
		Abort if $\mathsf{HD}(\mathsf{r}, \tilde{\mathsf{r}}) > \tau$

Fig. 1. Basic PUF-based unilateral authentication protocol.

III. PRELIMINARIES

This section provides an overview of a PUF-based authentication protocol, and the linear additive model that forms the basis of ML-based modeling attacks on PUF. A list of frequently used notations in this article is provided in Table I.

A. Basic SPUF-Based Authentication Protocol

The basic SPUF-based authentication method for unilateral authentication involves a verifier, usually a server, and a prover, which is a device embedded with only a single SPUF. The main operation is triggered by the server sending a challenge $\boldsymbol{c}$ to request for an authentication of the device. The device responds by activating its embedded SPUF to generate a response $\boldsymbol{r}$ to the server for verification. The complete process for this basic SPUF authentication protocol is depicted in Fig. 1. Before a device i is deployed, it will undergo a one-time enrollment process in a secure and control environment whereby d CRPs,[1] (c_{ij}, r_{ij}) (j is the jth CRP and $j \in [1\ d]$) are collected from the device and stored along with the device identifier (ID) in the server secure database. The device ID needs not be kept secret, and can be stored in an on-chip one-time programmable (OTP) memory. Upon device deployment, for each authentication process, a challenge $\boldsymbol{c}$ is sent to the device i, and a response $\tilde{\boldsymbol{r}}$ generated by an SPUF, is returned to the server. Authentication fails if the HD between the enrolled response $\boldsymbol{r}$ and the device response $\tilde{\boldsymbol{r}}$ exceeds an acceptable noise threshold τ computed based on the reliability of the SPUF. The authenticated CRP is discarded after use to avoid a further authentication event from being replayed by an adversary. The protocol is assumed to be server initiated typically for two reasons. First, as the server is a master that serves multiple slave devices, this can avoid a denial-of-service (DoS) attack from preventing the master to serve other slave devices. Second, the server is assumed to be resource rich. The same protocol that is server initiated can be easily adapted to device initiated if necessary.

[1] d should be sufficiently large to cater for all the authentication events throughout the service life of the device.

B. Linear Additive APUF Model

APUF [23] is one of the most widely studied SPUFs used in the above-mentioned authentication protocol. It consists of two parallel n-stage multiplexer (MUX) chains that feed into an arbiter stage to produce one response bit from an n-bit challenge, $c_0, c_1, \ldots, c_{n-1}$.

It has been shown that an APUF can be modeled by a linear additive model since its response bit to an input challenge can be derived by summing the delay difference in each stage. This model has been used by several ML methods, e.g., [12]–[15], to successfully attack practical delay-based SPUFs that are composed of linearly cascaded bit-slice circuits. Based on the additive delay model [6], the final delay difference between the top and bottom paths of the APUF, denoted by $\Delta(n)$, can be represented by

$$\Delta(n) = \boldsymbol{P} \cdot \boldsymbol{\omega}^T \tag{1}$$

where $\boldsymbol{P} = (p_0, p_1, \ldots, p_n)$ is a parity vector and $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_{n+1})$ is a weight vector.

The elements of $\boldsymbol{\omega}$ are dependent on the multiplexer switching and routing delays. They are susceptible to manufacturing process variations even if the switch of each APUF stage is identically designed. The elements of $\boldsymbol{P}$, on the other hand, depend only on the challenge bits. If the delay difference at the output of the final stage, $\Delta(n)$, is greater than 0, the response bit $\boldsymbol{r}$ is 1; otherwise, it is 0. The response bits of such an SPUF design can be predicted by building a software clone using this linear additive delay model. To succeed, a subset of CRPs needs to be collected to learn the weight vector $\boldsymbol{\omega}$ using ML algorithms.

Evolutionary strategies (ES) are another powerful ML technique that uses primarily mutation and selection to model an SPUF. The main idea is to generate random PUF instances and find the instance that best matches the real PUF model iteratively. The reliability-based CMA-ES attack [14] utilizes repeated measurements to observe the reliability of the response bits. The reliability information is then fed into a fitness function to find the best fit delay parameters. It outperforms traditional modeling attacks on XOR APUF with a large number of XORed APUFs. It is demonstrated in [15] that even highly obfuscated responses of, e.g., Slender PUF [28] and reverse fuzzy extractor [25], can be attacked using CMA-ES.

In view of these advanced attacks, we propose a deception protocol that will immensely increase the time and cost of the adversary to successfully model an SPUF, and resist attacks that require repeated challenge–response measurements. Our adversary model assumes that the device's enrollment interface upon deployment and the communication channel of the authentication protocol are publicly accessible. In other words, the attacker can freely query the device with selected challenges, and eavesdrop, manipulate, and reply protocol messages between the server and the device.

IV. Proposed Deception Protocol

The problem with typical SPUF-based authentication protocol in Fig. 1 is the adversary can unrestrictedly query the device to derive $(\boldsymbol{c}, \tilde{\boldsymbol{r}})$ and use them for training. It is also impractical for the resource-limiting device to keep track of the used challenges or use encryption in the authentication protocol to prevent replay and MITM attacks.

In addressing these problems, we would like to leverage on recent compact authentication protocol concepts [9], [30], [31] but avoid certain pitfalls in these and existing SPUF-based authentication schemes reviewed in Section II. (1) A cheaper but safer way than using a monotonic counter to generate a device nonce is needed so that the device will never or rarely send a used challenge; (2) Either the challenge or response of the SPUF should be encoded or obfuscated and decodable without requiring expensive cryptographic primitives at the device side; (3) To have the full CRP set at the disposal of the server with a constant size storage but avoid weakening reliability and security of the underlying SPUF after enrollment; (4) Resistance against ML attacks should be enhanced without limiting the number and frequency of authentication requests by the genuine server.

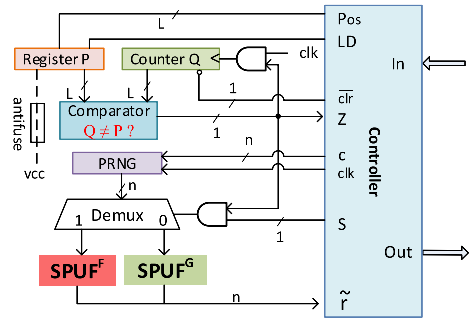


Fig. 2. Proposed lockdown encapsulation of genuine SPUF for deception-based authentication.

With these in mind, we propose an *active* deception technique to prevent the model-building attack on $\mathsf{SPUF}^{\mathsf{G}}$ by direct brute-force query on device enrollment interface. A timeout mechanism with the fake PUF multiplexing technique is incorporated into the challenge–response exchange protocol of Fig. 1 after the enrollment phase. A deception-based mutual authentication protocol is also introduced to enable timeout-free authentication by the server who has the real PUF model.

A. Device Architecture

The device-side supporting architecture consists of a monolithic encapsulation of essentially a genuine SPUF ($\mathsf{SPUF}^{\mathsf{G}}$), a fake SPUF ($\mathsf{SPUF}^{\mathsf{F}}$), a PRNG, a binary counter, a register, and a comparator, as shown in Fig. 2. The PRNG is typically implemented by a maximum length LFSR. It is used to generate random subchallenges from a seed challenge. The $\mathsf{SPUF}^{\mathsf{F}}$ is used to generate random nonce or fake responses. The counter, register, and comparator provide a practically realizable instantiation to turn a machine-learnable SPUF during enrollment prohibitively time consuming to learn upon deployment.

With reference to Fig. 2, when an n-bit input challenge c is applied to the device, m subchallenges are derived from the PRNG using c as a seed. These subchallenges are input to either $\mathsf{SPUF}^{\mathsf{G}}$ or $\mathsf{SPUF}^{\mathsf{F}}$ to generate an m-bit response. The time between two consecutive input challenges to the device can be tracked by a binary counter of length L. For an internal clock generator of period $\mathsf{T}_{\mathsf{clk}}$, the elapsed time is determined by the product of the counter output Q and $\mathsf{T}_{\mathsf{clk}}$. The counter Q is cleared upon power-up reset and whenever a response is sent externally (via $\overline{clr}$). The clock input to counter Q is gated by the comparator output Z. The comparator compares the output of the counter Q with the content of the register P and outputs $Z = 1$ if they are not equal, and outputs $Z = 0$ otherwise. The counter Q will count continuously until it is halted by $Z = 0$. The select enable strobe, $S = 1$ upon reset

Device i		Server
	$\xleftarrow{c_{ij}}$	$c_{ij} \leftarrow \mathsf{TRNG}(n)$
$\langle c_{ij} \rangle \leftarrow \mathsf{PRNG}(c_{ij})$		
$r_{ij} \leftarrow \mathsf{SPUF^G}(\langle c_{ij} \rangle)$		
$\overline{clr} \leftarrow 0$	$\xrightarrow{r_{ij}}$	
		Repeat for $j \in [1\ d_i]$
		Build $\mathsf{SPUF^G_{model}}$ with (c_{ij}, r_{ij})
$\mathsf{P_k} : S@1$		

Fig. 3. Enrollment protocol for model-based authentication.

so that the response to an input challenge c is determined by the comparator output Z. The response will be output from $\mathsf{SPUF^G}$ if $Z = 0$ and from $\mathsf{SPUF^F}$ if $Z = 1$. If $S = 0$, the true response from $\mathsf{SPUF^G}$ will always be output. The output of P is the sum of two components, $\mathsf{P_{min}}$ and $\mathsf{P_{os}}$. $\mathsf{P_{min}} = 2^k$ is fixed after the enrollment phase and $\mathsf{P_{os}}$ is doubled upon the output of every fake response. A simple controller is used to manage the inputs and outputs to the external world when the device is queried.

B. Enrollment

A model-based authentication approach is considered. To make the full CRP space of $\mathsf{SPUF^G}$ available at the server without incurring exponential time-space complexity to read and store them, the prerequisite is that $\mathsf{SPUF^G}$ must be learnable with polynomial resources in terms of training data and runtime. The enrollment process is performed once in a secure and controlled environment. The monolithic encapsulation is locked down to prevent the inputs and outputs of CRPs of $\mathsf{SPUF^G}$ and $\mathsf{SPUF^F}$ from being accessed without going through the control logic. To enable model-based authentication, the enrollment protocol shown in Fig. 3 is needed to collect the CRPs of $\mathsf{SPUF^G}$ for model building after the device is manufactured. During enrollment, a sufficiently large number d_j of CRPs are extracted from $\mathsf{SPUF^G}$ of device j to train a model $\mathsf{SPUF^G_{model}}$ by the ML algorithm for later reproduction of any CRPs of $\mathsf{SPUF^G}$ with a high accuracy. As the counter Q and the register P are both cleared to zero, and $S = 1$ upon reset, $Z = 0$ throughout the enrollment process. This will stop the counter and allow the response to be output from $\mathsf{SPUF^G}$ for each n-bit input challenge. After the authentication verification model $\mathsf{SPUF^G_{model}}$ has been successfully built and validated, one bit of register P is forced to stick at logic one permanently ($\mathsf{P_k} : S@1$) by an irreversible antifuse or by using an OTP nonvolatile memory (NVM) bitcell[2] to turn on the deception mechanism.

We assume a generic SPUF that has an acceptable response reliability for authentication application. If the raw responses of the SPUF do not meet the reliability expectation, several lightweight reliability enhancement techniques reported in the literature for commonly used SPUFs can be adopted. Examples of two widely adopted techniques with negligible overheads are spatial and temporal majority votings [40], [41]. The following additional measures are carried out during the enrollment phase to minimize the impact of SPUF reliability on the accuracy of $\mathsf{SPUF^G_{model}}$. First, the temperature and supply voltage used for collecting the CRPs for model building are well regulated and monitored. Second, the responses to the same challenges are collected multiple times and the majority voted CRPs are used for the training. Third, before shorted out the antifuse of P, $\mathsf{SPUF^G_{model}}$ is validated against the physical $\mathsf{SPUF^G}$ under varying operating conditions to determine the fractional HD (FHD) threshold τ required for authentication and the number of reauthentication required to reject or recall a device in the field.

Upon deployment, if the same protocol is used by the adversary to query the system, the response will only be output from $\mathsf{SPUF^G}$ if the challenge is input after a duration of $\mathsf{T_w} = \mathsf{T_{min}} + \mathsf{T_{os}}$ from system reset or from the output of the previous response. The period of $\mathsf{T_{clk}}$ to the L-bit counter has been set to approximately $\mathsf{T_{min}}/2^k$, where $k \in [0, L-1]$ is the position of the register output bit that has been forced into the stuck-at-one state after enrollment. Therefore, the range of waiting time $\mathsf{T_w}$ to apply the next challenge to $\mathsf{SPUF^G}$ can be varied from $\mathsf{T_{min}}$ to $\approx 2^{L-k} \times \mathsf{T_{min}}$ by loading the remaining $L - 1$ bits of the register with a nonzero integer. The maximum offset time $\mathsf{T_{os}}$ that can be added to $\mathsf{T_{min}}$ ranges from $2^{L-1} \times \mathsf{T_{clk}} \approx (1/2)\mathsf{T_{min}}$ for $k = L - 1$ (i.e., the MSB of the register) to $(2^L - 2) \times \mathsf{T_{clk}} = (2^L - 2) \times \mathsf{T_{min}}$ if $k = 0$ (i.e., the LSB of the register). The waiting time $\mathsf{T_w}$ can be changed by the server along with a bilateral authentication. Unlike existing model-based authentication protocol [9], this adaptive $\mathsf{T_w}$ allows the genuine server to make use of the direct query interface occasionally, for instance, to verify the $\mathsf{SPUF^G}$ of a recalled device or to retrain $\mathsf{SPUF^G}$ due to SPUF aging by resetting $\mathsf{T_w}$ to $\mathsf{T_{min}}$. Retraining an SPUF model usually takes a much lower number of CRPs than training a model from scratch. $\mathsf{T_{min}}$ can be set higher for an SPUF that requires less training CRPs and lower for those requires more training CRPs to minimize the time cost for these exceptional situations that are not related to regular field authentication. Prior to the redeployment of the SPUF in the field, the waiting time $\mathsf{T_w}$ for the attackers can be increased by the genuine server by setting $\mathsf{T_{os}}$ with a mutual authentication.

C. Mutual Authentication

The genuine server can authenticate the deployed device by using the mutual authentication protocol shown in Fig. 4. The genuine server can concurrently change the waiting time $\mathsf{T_w}$ for each successful authentication. This can prevent an adversary from being able to collect enough valid responses by a brute-force query through the enrollment protocol within a practical time limit.

To initiate a bilateral authentication, the server randomly selects an unused half-length challenge c_s and sends it to the device. Upon receiving c_s, the device uses it as a *seed* to its PRNG to generate $n/2$ random subchallenges $\langle c \rangle$. The device then clears the counter to ensure that $Z = 1$ so that the subchallenges $\langle c \rangle$ are applied to the fake SPUF ($\mathsf{SPUF^F}$) to produce a half-length challenge c_d to the server. The process is aborted by the server if the device ID is invalid or c_d has been used.

[2]OTP-based NVM with antifuse has been widely employed, e.g., the secret keys of automotive are stored in antifuse OTP to make them unattainable [38]. Embedded OTP NVM of high levels of security, high yields, low power, and excellent reliability in standard CMOS processes are available in Synopsys IP library [39].

Device i		Server
		do while (c_s used for ID$_i$)
$c_s \leftarrow \mathsf{TRNG}(n/2)$		
$\overline{clr} \leftarrow 0$	$\xleftarrow{c_s}$	
$S \leftarrow' 1'$		
$\langle c \rangle \leftarrow \mathrm{PRNG}(c_s)$		
$c_d \leftarrow \mathsf{SPUF^F}(\langle c \rangle)$	$\xrightarrow{\mathrm{ID}_i \,\|\, c_d}$	
		Abord if ID$_i$ is invalid or c_d has been used
		Add c_d into used c_d list of ID$_i$
		$\langle c \rangle \leftarrow \mathrm{PRNG}(c_s \,\|\, c_d)$
		$r_1 \,\|\, r_2 \,\|\, r_3 \leftarrow \mathsf{SPUF^G_{model}}(\langle c \rangle)$
		$\mathsf{P_{os}} \leftarrow \mathsf{TRNG}(L-1)$
		$h_1 \leftarrow \mathsf{P_{os}} \oplus r_1$
		$h_2 \leftarrow \mathsf{P_{os}} \oplus r_2$
	$\xleftarrow{h_1 \,\|\, h_3}$	$h_3 \leftarrow r_1 \oplus r_3$
$\langle c \rangle \leftarrow \mathrm{PRNG}(c_s \,\|\, c_d)$		
$S \leftarrow' 0'$		
$\widetilde{r_1} \,\|\, \widetilde{r_2} \,\|\, \widetilde{r_3} \leftarrow \mathsf{SPUF^G}(\langle c \rangle)$		
$\widetilde{h_3} = \widetilde{r_1} \oplus \widetilde{r_3}$		
if $\mathrm{FHD}(\widetilde{h_3}, h_3) \le \tau$,		
$\{\widetilde{\mathsf{P_{os}}} \leftarrow h_1 \oplus \widetilde{r_1}$		
load $\widetilde{\mathsf{P_{os}}}$ into register P,		
$\widetilde{h_2} \leftarrow \widetilde{P_{os}} \oplus \widetilde{r_2}\}$		
else $\{S \leftarrow' 1'$,		
$\widetilde{h_2} \leftarrow \mathsf{SPUF^F}(\langle c \rangle)\}$		
$\overline{clr} \leftarrow 0, S \leftarrow' 1'$	$\xrightarrow{\widetilde{h_2}}$	
		if $d = \mathrm{FHD}(\widetilde{h_2}, h_2) > \tau$, abort

Fig. 4. Proposed deception-based mutual authentication protocol flow.

Otherwise, the server concatenates c_s and c_d to complete the full-length challenge $c_s||c_d$. This full-length challenge is used as a *seed* to the same PRNG as the device to generate $3m$ subchallenges $\langle c \rangle$. These subchallenges are used to produce three m-bit responses, r_1, r_2, and r_3 from $\mathsf{SPUF^G_{model}}$. Two helper data, $h_1 = \mathsf{P_{os}} \oplus r_1$ and $h_3 = r_1 \oplus r_3$, are computed, where $\mathsf{P_{os}} = \mathsf{T_{os}}/\mathsf{T_{clk}}$ is a $(L-1)$-bit positive integer. If $m \ge L$, the m-bit response is truncated to match the length of $\mathsf{P_{os}}$ before the XOR operation. h_1 and h_3 are concatenated into $h_1||h_3$ and sent to the device.

Upon receiving the $(L+m)$-bit string, using the same subchallenges $\langle c \rangle$ with $c_s||c_d$ from its PRNG, the device generates three m-bit responses, $\widetilde{r_1}$, $\widetilde{r_2}$, and $\widetilde{r_3}$ by setting $S = 0$ to select $\mathsf{SPUF^G}$ for the application of $\langle c \rangle$. Then, it computes $\widetilde{h_3} = \widetilde{r_1} \oplus \widetilde{r_3}$. If $\widetilde{h_3}$ matches the received h_3 within acceptable FHD tolerance, the server is authenticated. The device then recovers $\widetilde{\mathsf{P_{os}}}$ from the received h_1 by XORing it with $\widetilde{r_1}$.

The recovered $\widetilde{\mathsf{P_{os}}}$ is loaded into the register to change the waiting time to $\mathsf{T_w} = \mathsf{T_{min}} + \widetilde{\mathsf{T_{os}}}$. The device will acknowledge the successful update by sending $\widetilde{h_2}$ to the server, where $\widetilde{h_2} = \widetilde{\mathsf{P_{os}}} \oplus \widetilde{r_2}$. Otherwise, if the FHD between $\widetilde{r_3}$ and r_3 exceeds the acceptable tolerance, the device sets $S = 1$ to generate $\widetilde{h_2} = \mathsf{SPUF^F}(\langle c \rangle)$ and sends it to the server. The counter Q is cleared by setting $\overline{clr}$ upon transmission of $\widetilde{h_2}$. The server can verify the authenticity of the device by checking the received $\widetilde{h_2}$ against $h_2 = \mathsf{P_{os}} \oplus r_2$. If they are equal within an acceptable FHD tolerance, the device is authenticated successfully. Otherwise, the authentication fails and the process is aborted.

It should be noted that the adversary cannot issue an unseen packet $h_1||h_3$ that has not been issued by the server to obtain the corresponding response packet $\widetilde{h_2}$. Consequently, the uniqueness of every authentication is assured by the fresh challenge. Because r_1, r_2, and r_3 are generated from the same fresh starting challenge and "locked" to each other by design, replaying either half challenge will not produce new response bits. The random P_{os} helps to conceal r_1 and r_2, and need not be precisely tracked. Even if $\widetilde{r_1} \ne r_1$ due to the reliability of $\mathsf{SPUF^G}$, as long as the error due to the recovered $\widetilde{P_{os}}$ and the response $\widetilde{r_2}$ generated by $\mathsf{SPUF^G}$ does not cause the FHD between $\widetilde{h_2}$ against h_2 to exceed τ, the outcome of the authentication will not be affected. Since we do not limit the number of challenges like [9], the genuine server can authenticate the same device again by this protocol with a fresh challenge before rejecting the device.

Device i		Imposter
	$\xleftarrow{c}$	Select c
$\langle c \rangle \leftarrow \mathsf{PRNG}(c)$		
if 'Z' $= 0, \tilde{r} \leftarrow \mathsf{SPUF^G}(\langle c \rangle)$		
if 'Z' $= 1, \{\tilde{r} \leftarrow \mathsf{SPUF^F}(\langle c \rangle), \mathsf{P_{os}} \leftarrow 2 \times \mathsf{P_{os}}\}$		
$\overline{clr} \leftarrow 0$	$\xrightarrow{\tilde{r}}$	

Fig. 5. Deception against brute-force model-building attack.

D. Deception-Based Technique

Although the direct query protocol of Fig. 3 is used only once during device enrollment by the genuine server, this direct query may also be used by the adversary upon device deployment. Fig. 5 shows the device's action when the enrollment protocol is utilized by the adversary to collect the CRPs for model building.

When the device is queried by a challenge c, the device reads the comparator output Z. If $Z = 0$, the controller will apply the subchallenges $\langle c \rangle$ derived from c to the genuine SPUF ($\mathsf{SPUF^G}$) to produce a response $\widetilde{r}$ to the query. If $Z \ne 0$, the response $\widetilde{r}$ will be output from the fake SPUF ($\mathsf{SPUF^F}$) by applying $\langle c \rangle$ to it. The device clears the counter Q by setting $\overline{clr} = 0$ and doubles $\mathsf{P_{os}}$ upon transmitting $\widetilde{r}$.

$Z = 0$ when the time lapse T of the current query from the last output response is at least $\mathsf{T_{min}}$. This frequency of authentication events is perceived to be normal. $Z \ne 0$ when $\mathsf{T} = \mathsf{Q} \times \mathsf{T_{clk}} < \mathsf{T_w}$, where $\mathsf{T_w} = \mathsf{P} \times \mathsf{T_{clk}}$. Therefore, $Z \ne 0$ signifies that the authentication events are unusually frequent. As each consecutive query within the waiting time $\mathsf{T_w}$ will double $\mathsf{P_{os}}$, the waiting time will extend rapidly. Hence, an adversary who uses the enrollment protocol to brute-force attack the SPUF system will receive fake responses from the $\mathsf{SPUF^F}$. Using the incorrect CRPs to train the model will result in either nonconvergence or convergence with highly inaccurate prediction results. This will be further demonstrated in Section V. Unknowingly using such an incorrect SPUF model to mount an attack on the target device will easily expose the adversary.

The appropriate value of $\mathsf{T_w}$ depends on the use case. Even with the same number of CRPs required for a successful attack on an SPUF, $\mathsf{T_{min}}$ and $\mathsf{T_{os}}$ can be initialized differently according to the application risk. Different applications have different service spans and diminishing the profitability of a successful attack with time while the device is still in operation. For example, $\mathsf{T_w}$ for a disposable near-field communication (NFC) wristband for a month or multiday convention can be shorter than that of an

RFID for supply chain tracking. The dynamic penalty component of the waiting time $\mathsf{P_{os}}$ is controllable by the genuine server. It helps to quickly contaminate the CRPs collected by the adversary who attempts to model $\mathsf{SPUF^G}$ in a practical amount of time. As opposed to alerting the attacker of an incorrect attempt by inaction, the interplay between false PUF multiplexing and timeout mechanism exponentially increases the attacker's time and exhausts the attacker's resources to build an incorrect model. It causes the attacker to miss the opportunity cost of attacking another target or changing their attack strategy for a better chance of success. A timely toy example is an electronic wristband used for contact tracing during the pandemic. Its frequency of use rules out authentication protocols against the ML attack that limit the number of authentications. Our proposed mutual authentication protocol will detect imposer who wears a counterfeit wristband, as the SPUF clone trained from multiplexed responses of $\mathsf{SPUF^G}$ and $\mathsf{SPUF^F}$ of a genuine wristband will fail the authentication.

E. Attack Scenarios

1) Replay or Spoofing Attack: The easy way of preventing replay attack is to ensure freshness of challenge so that the adversary is unable to produce a new response to any not yet used challenge. To establish a trust transmission of sensitive data from the sender to the receiver, it is sufficient to ensure that the used CRPs do not allow the adversary to successfully impersonate the recipient (prover) to obtain subsequent sensitive data transmitted from the sender (verifier). As long as the challenge is sent by the verifier and never been reused, then a replay attack cannot be used to impersonate the prover. For the mutual authentication protocol of Fig. 4, since the server can keep track of all the used half challenges c_d from the device, the adversary cannot impersonate the device by replaying any used half challenge c_d in response to c_s from the server. Neither can the adversary be able to produce a valid h_3 to an unseen new challenge $c_s||c_d$ or be authenticated as a legitimate server by replaying an eavesdropped $h_1||h_3$ to a new challenge $c_s||c_d$. Without the genuine device $\mathsf{SPUF^G}$ or its model $\mathsf{SPUF^G_{model}}$, it is impossible for the adversary to modify an intercepted c_s and c_d to match r_3. Due to the onewayness of $\mathsf{SPUF^G}$, it is also impossible to modify the intercepted $h_1||h_3$ to obtain new $c_s||c_d$, h_1, h_3, and $\widetilde{h_2}$ that will pass authentication at both sides. The server will be alerted of any abnormalities by its inability to receive an acceptable $\widetilde{h_2}$ in a given time after sending out $h_1||h_3$. Hence, it is impossible for the adversary to spoof the protocol by the MITM attack.

2) Strong Knowledge Attack: In the proposed deception-based mutual authentication protocol, neither the real responses, r_1, r_2, and r_3, to the challenge $c_s||c_d$ nor the threshold time offset parameter $\mathsf{P_{os}}$ are transmitted in clear between the server and the device. Without the genuine PUF or its model, the secrecy of r_1, r_2, and r_3 will not be compromised by divulging h_1, h_2, and h_3. The device authenticates the server by checking $\widetilde{h_3}$ computed from its generated $\widetilde{r_1}$ and $\widetilde{r_3}$ against h_3. It is computationally intractable for an adversary to determine a new $c_s'||c_d'$ with the correct r_1 and r_3 that will map to h_3 transmitted by the server without $\mathsf{SPUF^G_{model}}$. Even if this is possible, the adversary cannot force $\mathsf{SPUF^F}$ of the device to produce this specific c_d' from his chosen challenge c_s'.

The server authenticates the device by checking h_2 against $\widetilde{h_2}$ returned by the device. The probability of correctly guessing the output of XORing an unknown random binary bitstream and a known binary bitstream of length $L-1$ is 2^{1-L}. Since the adversary needs to correctly predict $\mathsf{P_{os}}$ from h_1 without knowing r_1 and then predict h_2 from $\mathsf{P_{os}}$ without knowing r_2, the probability of an adversary being successfully authenticated by the server is $2^{2(1-L)}$. By making the register length at least 64 bits, this probability is at most 2^{-126}.

3) Modeling Attack: The adversary needs to collect enough CRPs from a device to train a PUF model before it can be used to accurately predict any unused CRPs. In the proposed deception protocol, with the kth bit of P permanently stuck-at-one after enrollment, the adversary can only obtain one real SPUF response after a minimum waiting time of $\mathsf{T_w} = (2^k + \mathsf{P_{os}}) \times \mathsf{T_{clk}}$. The stuck-at-one bit position k of the register and the counter clock frequency can be fixed at design time. $\mathsf{T_{os}}$ can also be initialized so that $\mathsf{T_w}$ upon deployment can be made long enough to prevent the adversary from collecting enough valid CRPs within a practical time span through the direct query protocol. The offset component $\mathsf{T_{os}}$ of $\mathsf{T_w}$ is not static but changed randomly upon each successful authentication by the genuine server and doubled for every unsuccessful authentication attempt by the adversary. To be able to collect the CRPs from the $\mathsf{SPUF^G}$ at the shortest possible time, the adversary will have to first determine $\mathsf{T_w}$ by comparing the responses obtained from sending the same challenge at different time intervals apart. If $\mathsf{SPUF^F}$ is used to generate the fake responses, when the same challenge is sent consecutively within a short time, all responses are identically subjected to a small probability of error due to the reliability of $\mathsf{SPUF^F}$, which may allure the attacker to progressively extend the waiting interval to apply the same challenge until a different response is obtained. As each trial within the current $\mathsf{T_w}$ will double $\mathsf{T_{os}}$, the waiting time will blow up to months or years after just a few incorrect attempts depending on the $\mathsf{T_{os}}$ before the first incorrect guess. The adversary can hardly collect enough true CRPs within a practical amount of time to model $\mathsf{SPUF^G}$.

V. Results and Analyses

A. Test Setup for Modeling Attacks

CMA-ES attack is an effective ML-based modeling attack against PUF even in a blackbox setting [14], [15]. In particular, reliability-based CMA-ES attack demonstrates higher efficiency at breaking an l-XOR APUF than the LR attack, where it is comparatively more efficient when the number of parallel APUFs l is higher. Both CMA-ES and LR attacks are used to evaluate the proposed protocol. We follow the approaches in [12], [14], [15], [19], and [42] to perform the attacks on APUF. Specifically, we utilize the most recent published platform in [10] for the experiments in this work. The delay parameters, p_i, q_i, r_i, and s_i, of each stage in an APUF design described in Section II are randomly generated using a standard normal distribution $\sim \mathcal{N}(0, 1)$. The experimental results reported in [6] showed that 4.57% of noise were introduced into its response when the temperature was varied from 27 °C to 70 °C, and the responses were 2.16% noisier when the voltage was deviated from the rated voltage of 1.8 V by ±2%. To test the impact of noise on the proposed deception protocol, in all our experiments, a random variable $\sim \mathcal{N}(0, \sigma_{\text{noise}}^2)$ is inserted into each delay path, $\Delta(n)$, where $\sigma_{\text{noise}} = 0.5$ is derived from the practical noise level obtained in [6]. The challenges are generated randomly using Python. All CRPs are equally divided into two sets, a training set for modeling and a testing set for prediction.

The program of [42] is implemented to build an adversarial model for evaluating the effectiveness of false PUF multiplexing against the LR attack. In LR-based modeling attacks, the PUF is modeled using the response values. When a PUF cannot generate the same response all the time for a given challenge, the PUF response r can be modeled by (2) [20], where the delay difference $\Delta(n)$ is contributed by various sources of noise $\Delta_{\text{noise}}(n)$ in addition to the actual delay difference of the PUF $\Delta_{\text{actual}}(n)$

$$\Delta(n) = \Delta_{\text{actual}}(n) + \Delta_{\text{noise}}(n) = \boldsymbol{P} \cdot \boldsymbol{\omega}^T + \Delta_{\text{noise}}(n)$$
$$r = \begin{cases} 1, & \text{if } \Delta(n) > 0 \\ 0, & \text{if } \Delta(n) < 0. \end{cases} \tag{2}$$

This is used by the reliability-based CMA-ES attack [14] to find the best fit delay difference, $\boldsymbol{\omega}$, at each stage in (**??**). To compute the reliability H_i, the same challenge $\boldsymbol{c_i}$ is sent to the PUF t times. H_i is 1 if the t responses, $r_{i1}, r_{i2}, \ldots, r_{it}$, are all matched and 0 otherwise. The reliabilities of the responses to n different challenges are grouped into $\mathbf{H} = \{H_1, H_2, \ldots, H_n\}$. A group of p hypothetical reliabilities, $\widetilde{H_{i1}} \cdots \widetilde{H_{ip}}$, for the challenge $\boldsymbol{c_i}$ is then computed by testing all p possible absolute delay differences, $|\boldsymbol{P} \cdot \widetilde{\boldsymbol{\omega}^T}|$, generated by the CMA-ES algorithm. The fitness metric f_i used to evaluate the quality of the candidate solution $\widetilde{\boldsymbol{\omega}}$ is computed by the Pearson correlation coefficient between $\mathbf{H}_i$ and $\widetilde{\mathbf{H}}_{ij}$. However, this reliability-based CMA-ES attack does not help to accelerate or improve the accuracy of attacking our proposed deception protocol. It is difficult to determine the actual reliability $\mathbf{H}$ with the fake responses or a mixture of real and fake responses. As it requires applying the same challenge multiple times to obtain the true response reliability data for each challenge, more time is required to collect the CRPs even if the static $\mathsf{T_{min}}$ of $\mathsf{T_W}$ is known by the attacker. $\mathsf{T_W}$ has a dynamic component $\mathsf{T_{OS}}$ that is changeable by the server. Due to its multiplying effect to penalize the wrong prediction, if the attacker is misled by using the same waiting time predicted upfront to collect the training data, the repeated challenges will be worse off as most data collected after the first incorrect waiting time are likely to be fake. For this reason, only the original CMA-ES attack will be considered. Its fitness function is given by

$$f_i = \max_{j=1,\ldots,p} \left\{ \rho\left(\mathbf{R}_i, \widetilde{\mathbf{R}_{ij}}\right) \right\} \tag{3}$$

where $\mathbf{R}_i$ is the response obtained from the training set and $\widetilde{\mathbf{R}}_{ij}$ is the response generated by the CMA-ES algorithm. The higher the correlation coefficient between $\mathbf{R}_i$ and $\widetilde{\mathbf{R}}_{ij}$, the larger the f_i. The fitness metric f used to evaluate the best candidate $\widetilde{w}$ is derived by adding up all f_i for all the CRPs. The larger the value of f, the more accurate the PUF model.

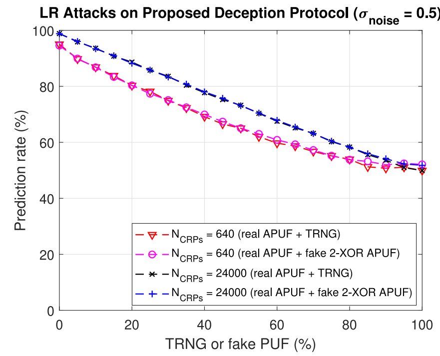


Fig. 6. Comparison of LR attack results for the proposed deception protocol utilizing either a TRNG or an XOR PUF as fake response generator. The y-axis shows the achieved correct prediction rate P_{pred} of the LR attacks based on different percentages of fake information mixed with the training responses.

A common adversarial model [14] is used for the evaluation. The program used to mount the CMA-ES attack is executed in MATLAB R2016b, by adopting the code for the core CMA-ES algorithm from [43]. For SPUF design such as APUF, one m-bit response is derived from m n-bit subchallenges generated from a PRNG to form one CRP, where m is the bit length of a response and n is the bit length of a challenge or the number of stages of an APUF. $k \times m$ n-bit subchallenges are used to produce k m-bit responses.

B. Effect of False PUF Multiplexing

The fake response generator is implemented by an SPUF design with a different circuit architecture from the genuine SPUF to produce vastly different CRPs. A TRNG [44] is implemented to generate random responses for comparison. The resistance of the deception protocol $\mathsf{SPUF^G} + \mathsf{SPUF^F}$ is evaluated by both LR and CMA-ES attacks and compared against that of $\mathsf{SPUF^G}$ + TRNG in this section.

1) Percentage of Valid/Invalid Responses on LR Attack: Two conventional APUFs, one for $\mathsf{SPUF^G}$ and the other for $\mathsf{SPUF^F}$, are employed to produce the CRPs for training. Three different sizes of CRP sets, $N_{\text{CRP}} = 640$ and $N_{\text{CRP}} = 24\,000$ (similar to [12] and [42]) are used for training. Depending on the percentage of fake information, x_f ($x_f \in \{0, 100\}$), a group of mixed response bits from both $\mathsf{SPUF^G}$ and $\mathsf{SPUF^F}$/TRNG is derived and used for the LR attack.

Fig. 6 depicts the LR attack results of the proposed deception protocol by using either $\mathsf{SPUF^F}$ or TRNG as its fake response generator. A varying number of fake CRPs is produced according to x_f. The prediction rate of a modeling attack is calculated by $P_{\text{pred}} = (N_{\text{correct}}/N_{\text{total}}) \times 100\%$, where N_{correct} and N_{total} are the number of correctly predicted response bits and the total number of response bits, respectively. For a random guess of a binary variable, the correct prediction of zero and one should be equally probable. Hence, the worst result is $P_{\text{pred}} = 50\%$. It can be seen that P_{pred} decreases proportionally with the percentage of fake responses x_f in the training samples. For the same x_f, the prediction rate is only slightly higher for a larger number of training samples, e.g., 24 000, than a smaller number, e.g., 640. For the same number of training CRPs and the same percentage of fake responses, the prediction rates are the same for both fake APUF and TRNG.

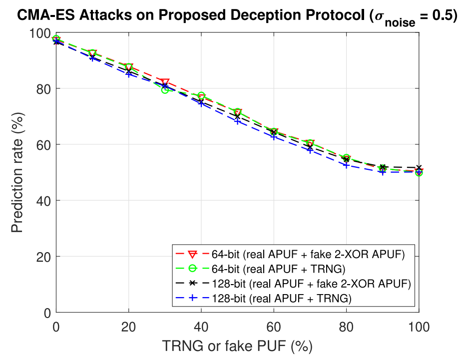


Fig. 7. CMA-ES attack results for the proposed deception protocol by applying different challenge bit lengths, 64-bit and 128-bit, as well as utilizing different fake response generators (TRNG and XOR PUF). The number of training samples used for this experiment, $N_{\text{CRP}} = 4000$, is the same as that used in [14].

2) Percentage of Valid/Invalid Responses on CMA-ES Attack: The CRPs ($N_{\text{CRP}} = 4000$) used for training consist of a mixed combination of responses collected from the real PUF and fake PUF/TRNG designs, depending on the percentage of fake responses. The proposed deception protocol is evaluated for two different challenge bit lengths, 64 and 128 bits.

Fig. 7 shows the CMA-ES attack results on the proposed deception protocol for different percentages of responses from the fake PUFs or TRNGs with different challenge bit lengths. The prediction rates for both lengths of challenges decrease proportionally with x_f from approximately 100% with all real responses to approximately 50% with 90% of responses generated from the fake PUFs/TRNGs in the 4000 training samples. It can be seen that P_{pred} decreases proportionally with the percentage of fake responses x_f in the training samples. Moreover, there is no appreciable difference in the prediction rates of the CMA-ES attack by using either a fake APUF or a TRNG to generate the fake responses for the proposed protocol.

From the above experiments, the percentage of fake responses has a greater impact on the prediction rate of both CMA-ES and LR attacks. Since the fake APUF or TRNG may generate the same response as that of the real APUF, we also evaluate the prediction rate for a given percentage of erroneous CRPs, i.e., a given percentage of errors occurred in the responses of training samples. Fig. 8 illustrates the impact on the prediction rate of the LR attack by using varying percentages of erroneous responses (without adding extra noise) of the real APUF for training. The erroneous responses for real APUF is generated by flipping its response bits. If a fake APUF is obtained from a real APUF with errors, it is more difficult for the attackers to predict the real APUF response using an LR attack. The results show that the correct prediction rate under this scenario drops rapidly to about 50% with slightly more than 50% bit error rate.

C. Effect of Dynamic and Static Components of Waiting Time on Modeling Attack

For every authentication attempt made by the adversary before the waiting time $\mathsf{T_w}$, the offset time $\mathsf{T_{os}}$ will be doubled. Hence, the waiting time after applying N_f challenges

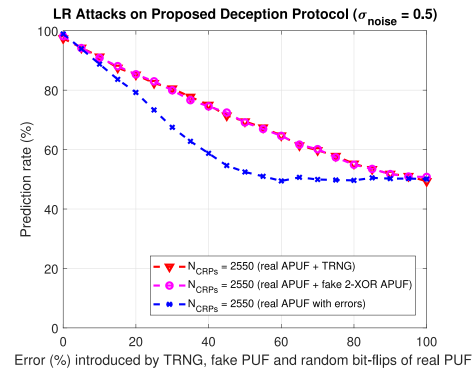


Fig. 8. LR attack results for the proposed deception protocol in mixing different fake information, including responses from fake APUF, TRNG, and real APUF with error injection.

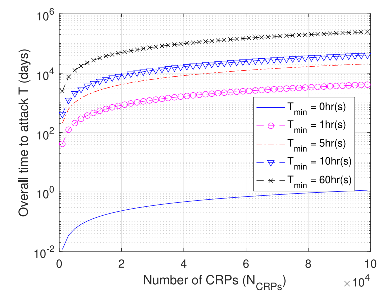


Fig. 9. Overall time T taken by modeling attacks to predict a 64-bit APUF with respect to the minimum number of training samples N_{CRP} and static $\mathsf{T_{min}}$ of the proposed deception protocol. The typical training time T_{train} is assumed to be 1 s.

with waiting time shorter than the current $\mathsf{T_w}$ is given by

$$\begin{aligned}\mathsf{T_w} &= \left(\mathsf{P_{min}} + \mathsf{P_{os}} + 2\mathsf{P_{os}} + \cdots + 2^{N_f}\mathsf{P_{os}}\right)\mathsf{T_{clk}}\\ &= \left\{2^k + \mathsf{P_{os}}(2^{N_f} - 1)\right\}\mathsf{T_{clk}}.\end{aligned} \tag{4}$$

For $\mathsf{T_{clk}} = 1$ ms, $L = 32$, $k = 21$, and $\mathsf{P_{os}} = 1000$, $\mathsf{T_{min}}$ is less than an hour. After only $N_f = 25$, the waiting time $\mathsf{T_w}$ exceeds 388 days. If $\mathsf{P_{os}}$ is updated by the server to 2^{25}, it takes only $N_f = 10$ adversarial attempts for the waiting time $\mathsf{T_w}$ to exceed 397 days. As analyzed in Section IV-E, the more the attacker attempts to close in their estimate of $\mathsf{T_w}$, the faster $\mathsf{T_w}$ will blow up to years. The attacker can derive a full set of valid training samples by always waiting for a time longer than $\mathsf{T_w}$ before sending the next challenge, provided that the genuine server did not authenticate the device to change the $\mathsf{P_{os}}$ during this period. If the attacker sends a challenge every t, where $t > \mathsf{T_w}$, the minimum overall time, T, to obtain the training set can be calculated by (5), which includes the time to derive all training samples, N_{CRP}, the training time for modeling the PUF, T_{train}, and the trial-and-error time for determining the $\mathsf{T_w}$, $T_{t\&e}$

$$T = N_{\text{CRP}} \times \mathsf{T_w} + T_{\text{train}} + T_{t\&e}. \tag{5}$$

Even if we omit $T_{t\&e}$ and assume that $\mathsf{T_w}$ is constant, the best attack time T without considering the delay penalty of $\mathsf{T_{os}}$ can still be made impractically long by varying $\mathsf{T_w}$ according to the number of training samples N_{CRP} required to successfully model the chosen genuine SPUF. To simulate the

TABLE II
LIST OF HEURISTIC MODELING ATTACK RESULTS FOR n-XOR APUF AND FEEDFORWARD APUF.
(TIME IN BRACKET REFERS TO ATTACK TIME IN YEARS WITH $\mathsf{T_{min}} = 1$ H AND $\mathsf{T_{os}} = 0$.)

PUF type	Evaluation	No. of stages	No. of XORs, n									
			1	4	5	6	7	8	9	10	16	32
n-XOR APUF (LR) [42]	Train time	64	0.13 sec. (0.3 yrs)	3.5 mins (1.3 yrs)	2.1 hrs (9.1 yrs)	1.29 days (23.0 yrs)	-	-	-	-	-	-
		128	0.5 sec. (0.64 yrs)	2.6 hrs (2.8 yrs)	16.3 hrs (57.2 yrs)	-	-	-	-	-	-	-
	No. of CRPs	64	2555	12,000	80,000	200,000	-	-	-	-	-	-
		128	5,570	24,000	500,000	-	-	-	-	-	-	-
n-XOR APUF (CMAES) [14]	Train time	128	0.9 hrs (2.3 yrs)	1.8 hrs (17.1 yrs)	-	-	-	3.3 hrs (34.3 yrs)	-	-	30.5 hrs (57.1 yrs)	60 hrs (228 yrs)
	No. of CRPs	128	20,000	150,000	-	-	-	300,000	-	-	500,000	2,000,000
Feed-Forward APUF(ES) [42]	Train time	64	-	-	-	7.51 mins (5.7 yrs)	47.07 mins (5.7 yrs)	47.07 mins (5.7 yrs)	47.07 mins (5.7 yrs)	47.07 mins (5.7 yrs)	-	-
		128	-	-	-	3.15 hrs (5.7 yrs)	3.15 hrs (5.7 yrs)	3.15 hrs (5.7 yrs)	3.15 hrs (5.7 yrs)	3.15 hrs (5.7 yrs)	-	-
	No. of CRPs	64	-	-	-	50,000	50,000	50,000	50,000	50,000	-	-
		128	-	-	-	50,000	50,000	50,000	50,000	50,000	-	-

optimistic attack time for this scenario in subsequent experiments of this section, instead of fixing $\mathsf{T_{min}}$ and changing $\mathsf{T_{os}}$ to achieve a given $\mathsf{T_w}$, we set $\mathsf{T_{os}} = 0$ and change $\mathsf{T_{min}}$ to avoid the multiplying delay penalty for incorrect waiting time. Under this premise, Fig. 9 shows that the efficiency of modeling attacks on existing unfortified ($\mathsf{T_w} = 0$ h) 64-stage APUF reduces with different $\mathsf{T_{min}}$. For example, a 64-stage APUF design can be predicted with 95% accuracy in approximately 0.01 s (the time for sending and reading one CRP is neglected) with a training sample size of $N_{\text{CRP}} = 640$ using the above LR-based experiment. Then using the proposed deception protocol with a threshold time of 1 h, it will require 25.2 days for the same N_{CRP} to achieve the same prediction rate. Similarly, the overall attack time T is also longer when more training samples N_{CRP} are needed if a larger APUF with more number of stages or a more complex XOR-PUF is used. The original attack time increases to 0.13 s and the prediction rate hits 99% when the number of training samples required is increased to $N_{\text{CRP}} = 2555$. By using the same $\mathsf{T_{min}}$ of 1 h with our protocol, without compromising the success rate of prediction, the overall attack time T will increase to 109 days. If $\mathsf{T_{min}}$ is set to 24 h, the attacker will require approximately 7.1 years to collect $N_{\text{CRP}} = 2555$ from a 64-stage APUF to achieve a prediction rate of 99%.

Table II shows the original training time and brute-force attack time with known $\mathsf{T_{min}}$ of 1 h and zero $\mathsf{T_{os}}$. They are computed based on the number of CRPs required by LR/ES modeling attack on n-XOR APUF and feedforward APUF[42], as well as CMA-ES attack on n-XOR APUF [14]. The original training time of modeling attack for a 128-stage feedforward arbiter PUF is approximately 3.15 h, but increases to almost $[50\,000/(24 \times 365)] = 5.7$ years by merely delaying the CRP collection with $\mathsf{T_{min}}$ of 1 h. The original training time for the 128-stage 8-XOR APUF is 3.3 h, but the attack time increases to 34.3 years with $\mathsf{T_{min}}$ of 1 h.

VI. HARDWARE IMPLEMENTATION

To demonstrate the proposed deception protocol using FPGA, it is important to choose an FPGA-based APUF design that has a high uniqueness and reliability. To this end, the lightweight flip-flop-based arbiter PUF (FF-APUF) design [45] is adopted since it has a higher uniqueness (∼40%) compared to the conventional APUF (∼9%) on Xilinx Artix-7 FPGA implementation. Moreover, a 64-stage FF-APUF achieves good reliabilities of 97.10% and 93.90% over a temperature range of 0 °C–70 °C and ±10% voltage variations, respectively. The uniformity of the FF-APUF design is ∼47%. It uses 44 slices, 130 look up tables (LUTs) and 172 flip flops (FFs).

TABLE III
HARDWARE RESOURCES OF 64-BIT FF-APUF, APUF, AND ANCILLARY COMPONENTS ON XILINX ARTIX-7 FPGA FOR THE PROPOSED DECEPTION PROTOCOL

Type	FF-APUF	LFSR	APUF/(TRNG)**	Counter	UART	Others*	Total	Percentage (%)
LUTs	130	65	134/(164)	28	51	170	578	0.91%
FFs	172	67	2/(139)	63	54	426	784	0.62%
Clk Cycles	192	132	132/(1)	6	-	-	324	-
Slices	44	25	134/(58)	21	23	141	388	2.45%

*Controller + Comparator + 64-bit Register + glue logic.
**One of two options, APUF or TRNG, for fake response generation.
The available hardware resources on Xilinx Artix-7 FPGA: LUT(63,400), FF(126,800), slice(15,850).

From the analysis of Section V-B, the fake response generator implemented by another SPUF produces no significant difference in ML attack results as a TRNG fake response generator. Since uniqueness is not a major concern for the fake response generator, a 64-bit APUF design is considered here so that there is no difference in the number of cycles required to generate a response bit from the application of an input challenge between the real and fake SPUFs. This will prevent the attacker from exploiting the timing difference to discriminate the responses between $\mathsf{SPUF^F}$ and $\mathsf{SPUF^G}$. The APUF consumes only 134 slices, 134 LUTs, and 2 FFs. The multiplexers in both FF-APUF and APUF are placed and routed by minimizing the skew between the top and bottom delay paths. To generate a 64-bit response from a 64-bit challenge, a 64-bit maximum length LFSR with a feedback polynomial of $x^{64} + x^{63} + x^{61} + x^{60} + 1$ is used to generate 64 random internal challenges with the input challenge as seed. The number of slices required to implement this LFSR is 25.

Table III shows the hardware resource consumption of the device with 64-bit CRPs for the proposed deception protocol

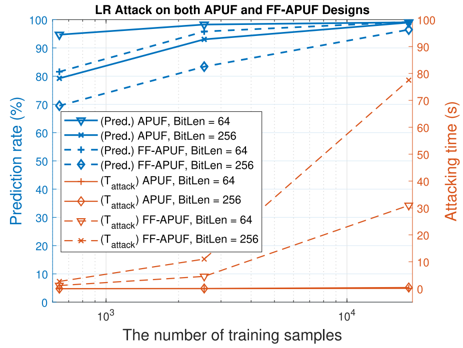


Fig. 10. Comparison of LR attack on both conventional APUF and FF-APUF designs. "BitLen" represents the number of stages of the real PUF.

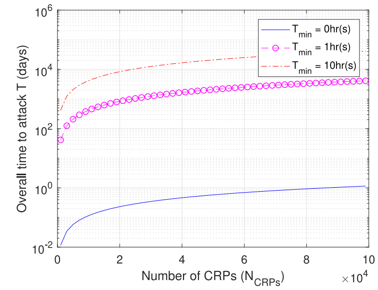


Fig. 11. Overall attack time T taken by modeling attacks to predict FF-APUF designs with respect to the minimum number of training samples N_{CRP} and static $\mathsf{T}_{\mathsf{min}}$ of the proposed deception protocol.

in terms of the numbers of LUTs, registers, clock cycles, and slices on the Xilinx Artix-7 FPGA. The clock frequency is set to 100 MHz. The propagation delays of FF-APUF and APUF can both be completed well within one clock cycle, i.e., 10 ns. Three clock cycles are consumed to load a 64-bit challenge, generate a 1-bit response, and store it into a register. To generate a 64-bit response, 3×64 clock cycles are required. FF-APUF occupies much fewer configurable fabrics of Xilinx Artix-7 FPGA than Slender PUF [28] and System-of-PUF [46] designs, which use 128 LUTs and 130 LUTs, respectively, for generating a 1-bit response. In total, 388 slices are consumed, which is only 2.45% of the resources of Xilinx Artix-7 FPGA. From Table III, the total time taken from authentication request to response generation by the device is $10\ \mathsf{ns} \times 264 = 2.64\ \mu$s, which is fast enough for most authentication protocols. The speed bottleneck of the authentication event is due to the data rate used for the serial communication between the server and the device. For our prototype experiment, this is limited by the UART with a baud rate of 115 200 bps for communication with the PC through RS232. It takes the device approximately $694\ \mu\text{s} \times 2 = 1388\ \mu$s to receive a 64-bit challenge and transmit a 64-bit response, including the start and stop bits required for the transmission of each byte. The total time for one authentication event is thus 1390.64 μs. This can be substantially reduced by using a faster serial link with a much higher baud rate. The total power consumption of the proposed protocol implemented on Xilinx Artix-7 FPGA is 101 mW.

Since TRNG can also be utilized for fake response generation, for comparison, we have implemented the coherent sampling ring oscillator-based TRNG (COSO-TRNG) from among the low-cost TRNGs suitable for FPGA implementation recently surveyed in [47]. As shown in Table III, this TRNG consumes 164 LUTs and 139 FFs, which are more than the 134 LUTs and two FFs of APUF.

Fig. 10 compares the prediction rate and attack time by the LR attack on using 64- and 256-stage FF-APUFs against conventional APUF of similar hardware complexity. For the same number of stages, FF-APUF requires significantly more number of training data to achieve the same prediction rate. Fig. 11 shows the attack time T for using FF-APUF as real APUF in our proposed deception protocol with different $\mathsf{T}_{\mathsf{min}}$. Compared with using conventional APUF for the same fixed $\mathsf{T}_{\mathsf{w}} = \mathsf{T}_{\mathsf{min}}$ with $\mathsf{T}_{\mathsf{os}} = 0$ in Fig. 9, T is comparable, if not slightly higher, for the same number of training samples N_{CRP}. Although the training time for each sample increases only negligibly, N_{CRP} increases significantly for the same prediction accuracy. Hence, the overall attack time increases substantially by replacing the conventional APUF with equally lightweight FF-APUF as $SPUF^{\mathsf{G}}$ in the proposed deception protocol.

The prediction rate of a 64-bit FF-APUF model trained with 18 050 CRPs during enrollment is 99.5%, as shown in Fig. 10. The worst reliability of the FF-APUF implemented on Xilinx Artix-7 FPGA is 93.9% at 10% underrated core voltage [36]. Since the average bit error rate due to the accuracy of FF-APUF model is negligibly small (0.5%) compared with the worst reliability of the physical FF-APUF in the field, the fraction of response bit differences between the model and the real FF-APUF will be kept below 7% for the device operating at temperature between 0 °C and 70 °C with no more than ±10% rated voltage variations. This is confirmed experimentally by the 94.17% prediction rate of the 64-bit FF-APUF model tested with 6% errors injected into the CRPs generated by the device PUF. Hence, the FHD tolerance τ can be conservatively set to 0.1 to reduce the false rejection rate to virtually zero. The barrier to modeling attack is still high as the model built by the attacker must have at least 90% accurate prediction rate for its response to be authenticated successfully.

VII. Protocol Comparison

The survey in [22] divides the PUF-based authentication protocols into two groups, *heavyweight* and *lightweight*. Most of the *heavyweight* protocols require either a strong cryptographic algorithm for privacy amplification and an ECC for response reconciliation [23]–[27]. For example, the *controlled PUF* [48] applies hashing to obfuscate its CRPs and requires ECC to correct the noisy responses. The ECC helper data has to be securely stored in order to prevent helper data manipulation attack [49]. A detailed review and comparison of the *heavyweight* authentication protocols have already been done in [22]. In this work, we focus on the *lightweight* group. Three *lightweight* protocols, including the *slender* PUF [28], *noise bifurcation* [29], and *system-of-PUF* protocols [46], as well as two recently published authentication protocols, *lockdown* [9] and *latent obfuscation* are considered.

Table IV compares the proposed deception protocol against the basic authentication protocol and five *lightweight* authentication protocols. *PUF-independent* relates to whether or not the protocol requires a specific PUF design; *Modeling resistance* refers to the protocol-assisted resistance to model

TABLE IV
COMPARISON OF THE PROPOSED DECEPTION PROTOCOL AGAINST LIGHTWEIGHT PROTOCOL FINALISTS FROM [22] AND TWO RECENT PUBLISHED PROTOCOLS, LOCKDOWN PROTOCOL FROM [9] AND LATENT OBFUSCATION [31]

Protocol	PUF-independent	Modeling resistance	Number of auth.	HW implement	TRNG
Basic authentication [5]	✓	×	d	✓	×
Slender PUF [28]	×	×	∞	✓	✓
Noise bifur. [29]	×	×	∞	✓	✓
Lockdown Ia [9]	✓	$\backsim$	d	$\backsim$	×
Lockdown Ib [9]	✓	✓	d	$\backsim$	×
Lockdown IIa [9]	✓	$\backsim$	d	$\backsim$	✓
Lockdown IIb [9]	✓	✓	d	$\backsim$	✓
Latent obfus. [31]	×	✓	d	$\backsim$	✓
System-of-PUF [46]	×	×	d	$\backsim$	×
Proposed protocol	✓	✓	∞	✓	×

The symbol × denotes 'no'. The symbol ✓ denotes 'yes'.
The symbol ∞ denotes the number of authentications is not preset but limited only by the CRP space of the SPUF.
The symbol d denotes the number of authentications has a prespecified limit.
The symbol $\backsim$ denotes result not available.

building attacks; *Number of auth.* refers to the quantity of CRPs that can be authenticated; *HW implement* evaluates the physical feasibility to implement these protocols on an FPGA or other hardware devices; and *TRNG* refers to any TRNG component used in these protocols.

As analyzed previously, the proposed deception protocol demonstrates good robustness to different ML-based attacks. Any type of SPUF can be used as the genuine SPUF or fake SPUF. Among the previous works, only *lockdown* protocols have no restriction on the PUF design. However, its implementation cost is not reported and its model-based authentication is designed based on system-level instantiation of hard-to-learn XOR-PUF instead of generic machine learnable SPUF.

Except *lockdown*, *latent obfuscation*, and our proposed deception protocols, other authentication protocols listed in Table IV are vulnerable to modeling attacks. Typically, a *TRNG* is used to generate a random substring to hide the response, e.g., in slender PUF [28]. In our work, a temporal control is used to delay the attacker from collecting enough correct CRPs from the real SPUF within a practical time. Overall, the proposed deception protocol is the only authentication protocol that has achieved all the desirable metrics. It is the only scheme that allows the practical realization of secure model-based authentication with generic machine-learnable SPUF. The number of authentications is limited only by the CRP space of the SPUF and the attacker is time restricted in making consecutive queries to collect the unused CRPs.

VIII. CONCLUSION

Security solutions today focus mostly on blocking attacks. Deception as a defense strategy provides greater delay, confusion, and disruption than rejecting sessions to the attacker's onslaught. It can drive preventive countermeasures to delay an attack, causing the adversary economic harm to figure out what is real and what is not, and hesitant to proceed. In this article, we propose a novel deception authentication protocol by deceiving the adversary to use a training set dominated by fake/invalid responses for ML. This will prevent their PUF clone from correctly predicting the response to the unknown challenge. The proposed deception-based challenge–response interface works with a real SPUF, a fake SPUF, a counter, a register, and a comparator to make the modeling of the real SPUF easy during enrollment but infeasible upon device deployment by delaying the collection of the required number of correct training data by an unpredictable and impractically long time. Attempts to shorten this data collection time will increase the number of fake CRPs used for training. The rapid drop of prediction accuracy with an increasing fraction of fake responses is demonstrated using two of the most widely known modeling attack techniques, LR and CMA-ES. The enrolled software model of real SPUF is used by the server to perform authentication requests through the proposed mutual authentication protocol. The protocol has been analyzed to be secure against replay and MITM attacks. The device-side components required to support the deception protocol are implemented on a Xilinx Artix-7 FPGA to validate its hardware efficiency. The entire authentication system for a prover consumes only 1.12% of LUTs and 0.62% of FFs. 388 slices are consumed overall, representing 2.45% of the resources of a Xilinx Artix-7 FPGA.

REFERENCES

[1] KrebsonSecurity. *DDoS on Dyn Impacts Twitter, Spotify, Reddit*. Accessed: Nov. 8, 2016. [Online]. Available: https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitter-spotify-reddit/

[2] C. H. Chang, Y. Zheng, and L. Zhang, "A retrospective and a look forward: Fifteen years of physical unclonable function advancement," *IEEE Circuits Syst. Mag.*, vol. 17, no. 3, pp. 32–62, Aug. 2017.

[3] C. Gu, J. Murphy, and M. O'Neill, "A unique and robust single slice FPGA identification generator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 1223–1226.

[4] C. Gu and M. O'Neill, "Ultra-compact and robust FPGA-based PUF identification generator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 934–937.

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

[6] 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. 2015.

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

[8] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, San Jose, CA, USA, Nov. 2008, pp. 670–673.

[9] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Jul.–Sep. 2016.

[10] 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," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 4, pp. 243–290, Aug. 2019.

[11] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2007, pp. 63–80.

[12] 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.

[13] 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. Identification Security Privacy Issues*, 2015, pp. 17–31.

[14] G. T. Becker, “The gap between promise and reality: On the insecurity of XoR arbiter PUFs,” in *Cryptographic Hardware and Embedded Syst. (CHES)*. Berlin, Germany: Springer, 2015, pp. 535–555.

[15] G. T. Becker, “On the pitfalls of using arbiter-PUFs as building blocks,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1295–1307, Aug. 2015.

[16] Q. Ma, C. Gu, N. Hanley, C. Wang, W. Liu, and M. O’Neill, “A machine learning attack resistant Multi-PUF design on FPGA,” in *Proc. 23rd Asia South Pac. Des. Autom. Conf. (ASP-DAC)*, Jeju Island, South Korea, Jan. 2018, pp. 97–104.

[17] S. Tajik *et al.*, “Physical characterization of arbiter PUFs,” in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, 2014, pp. 493–509.

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

[19] G. T. Becker *et al.* “Active and passive side-channel attacks on delay based PUF designs,” *Proc. IACR Cryptol. ePrint Archive*, vol. 2014, 2014, p. 287.

[20] J. Delvaux and I. Verbauwhede, “Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise,” in *Proc. Int. Conf. Hardw. Orient. Security Trust (HOST)*, 2013, pp. 137–142.

[21] J. Delvaux and I. Verbauwhede, “Fault injection modeling attacks on 65 nm arbiter and RO sum PUFs via environmental changes,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1701–1713, Jun. 2014.

[22] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, “A survey on lightweight entity authentication with strong PUFs,” *ACM Comput. Surveys*, vol. 48, no. 2, pp. 1–42, Oct. 2015.

[23] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. ACM 9th Int. Conf. Comput. Commun. Security (CCS)*, 2002, pp. 148–160.

[24] S. Katzenbeisser *et al.*, “Recyclable PUFs: Logically reconfigurable PUFs,” in *Cryptographic Hardware and Embedded Systems (CHES)*, B. Preneel and T. Takagi, Eds. Berlin, Germany: Springer, 2011, pp. 374–389.

[25] A. Van Herrewege *et al.*, “Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs,” in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Germany: Springer, 2012, pp. 374–389.

[26] Ü. Kocabaş, A. Peter, S. Katzenbeisser, and A.-R. Sadeghi, “Converse PUF-based authentication,” in *Proc. Int. Conf. Trust Trustworthy Comput.*, 2012, pp. 142–158.

[27] Y. Jin, W. Xin, H. Sun, and Z. Chen, “PUF-based RFID authentication protocol against secret key leakage,” in *Proc. Asia–Pac. Web Conf.*, 2012, pp. 318–329.

[28] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, “Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching,” in *Proc. IEEE Symp. Security Privacy Workshops*, May 2012, pp. 33–44.

[29] M. D. Yu, D. M’Raihi, I. Verbauwhede, and S. Devadas, “A noise bifurcation architecture for linear additive physical functions,” in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, May 2014, pp. 124–129.

[30] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba, “Applications of high-capacity crossbar memories in cryptography,” *IEEE Trans. Nanotechnol.*, vol. 10, no. 3, pp. 489–498, May 2011.

[31] 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. [Online]. Available: arXiv:1706.06232.

[32] J. Delvaux, “Attacks on the PUF-based authentication protocols YeHL16 and GaoMAAR17,” Cryptol. ePrint Archive, Rep. 2017/1134, 2017. [Online]. Available: https://eprint.iacr.org/2017/1134

[33] M. Barbareschi, A. De Benedictis, E. La Montagna, A. Mazzeo, and N. Mazzocca, “A PUF-based mutual authentication scheme for cloud-edges iot systems,” *Future Gener. Comput. Syst.*, vol. 101, pp. 246–261, Dec. 2019.

[34] M. Adeli and N. Bagheri, “Cryptanalysis of two recently proposed puf based authentication protocols for IoT: Phemap and salted phemap.” in *Proc. IACR Cryptol. ePrint Arch.*, vol. 2019, 2019, p. 1461.

[35] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, “A multiplexer-based arbiter puf composition with enhanced reliability and security,” *IEEE Trans. Comput.*, vol. 67, no. 3, pp. 403–417, Mar. 2018.

[36] 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.

[37] C. Gu, C. H. Chang, W. Liu, S. Yu, Q. Ma, and M. O’neill, “A modeling attack resistant deception technique for securing PUF based authentication,” in *Proc. Asian Hardw. Orient. Security Trust Symp. (AsianHOST)*, Dec. 2019, pp. 1–6.

[38] P. Piacentini. (Jun. 2017). *The Black Box in Auto Vehicles*. [Online]. Available: https://semiengineering.com/black-box-auto-vehicles/

[39] Synopsys. (Jan. 2019). *DesignWare OTP NVM IP*. [Online]. Available: https://www.synopsys.com/dw/ipdir.php?ds=nvm_otp

[40] P. Koeberl, J. Li, and W. Wu, “A spatial majority voting technique to reduce error rate of physically unclonable functions,” in *Trusted Systems*, R. Bloem and P. Lipp, Eds. Cham, Switzerland: Springer Int., 2013, pp. 36–52.

[41] A. Mills, S. Vyas, M. Patterson, C. Sabotta, P. Jones, and J. Zambreno, “Design and evaluation of a delay-based fpga physically unclonable function,” in *Proc. IEEE 30th Int. Conf. Comput. Design (ICCD)*, Sep. 2012, pp. 143–146.

[42] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” in *Proc. 17th Conf. Comput. Commun. Security (CCS)*, Oct. 2010, pp. 237–249.

[43] N. Hansen, “The CMA evolution strategy: A tutorial,” 2016. [Online]. Available: arXiv:1604.00772.

[44] F. Mei, L. Zhang, C. Gu, Y. Cao, C. Wang, and W. Liu, “A highly flexible lightweight and high speed true random number generator on FPGA,” in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 399–404.

[45] C. Gu, Y. Cui, N. Hanley, and M. O’Neill, “Novel lightweight FF-APUF design for FPGA,” in *Proc. IEEE 29th Int. Conf. System Chip (SOCC)*, Sep. 2016, pp. 75–80.

[46] S. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. Wong, “System-of-PUFs: Multilevel security for embedded systems,” in *Proc. IEEE Int. Conf. Hardw. Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2014, pp. 1–10.

[47] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, “A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices,” in *Proc. 26th Int. Conf. Field Program. Logic Appl. (FPL)*, Aug. 2016, pp. 1–10.

[48] B. Gassend, M. V. Dijk, D. Clarke, E. Torlak, S. Devadas, and P. Tuyls, “Controlled physical random functions and applications,” *ACM Trans. Inf. Syst. Security*, vol. 10, no. 4, p. 3, 2008.

[49] 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.

Chongyan Gu (Member, IEEE) received the Ph.D. degree from Queen’s University Belfast, Belfast, U.K., in 2016.

She is currently a Lecturer (Assistant Professor) with the School of EEECS, Queen’s University Belfast, where she is also a member of the Center for Secure Information Technologies, Institute of Electronics Communications and Information Technologies. She is an expert in hardware security. Her research into physical unclonable function (PUF) has been utilized as part of a security architecture for electronic vehicle charging systems, licensed by LG-CNS, South Korea, and was also licensed for evaluation by Thales, U.K. Her current research interests include PUFs, security in/for approximate computing, true random number generator, hardware Trojan detection, and machine learning attacks.

Dr. Gu has successfully organized two conference special sessions (IEEE APCCAS in 2018 and ACM GLSVLSI in 2020). Her team was the overall winner of INVENT 2015, a competition to accelerate the commercialization of innovative ideas. She was invited to give tutorial/talks to international conferences, such as IEEE ASP-DAC 2020 on the topic of practical PUF design on FPGA.

Chip-Hong Chang (Fellow, IEEE) received the B.Eng. degree (Hons.) from the National University of Singapore, Singapore, in 1989, and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

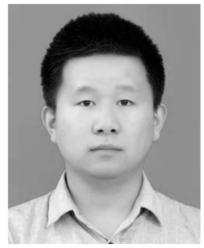
He joined the School of Electrical and Electronic Engineering (EEE), NTU, in 1999, where he is currently an Associate Professor. He held joint appointments with NTU as the Assistant Chair of Alumni of the School of EEE from 2008 to 2014, the Deputy Director of the Center for High-Performance Embedded Systems from 2000 to 2011, and the Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited five books, published 13 book chapters, more than 100 international journal papers (>70 are in IEEE) and more than 180 refereed international conference papers (mostly in IEEE), and delivered over 40 colloquia. His research interests include hardware security and trustable computing, deep learning network security, digital image processing algorithms and architectures for emerging vision sensors, physical layer watermarking, and residue number systems.

Dr. Chang currently serves as the Senior Area Editor for IEEE Transactions on Information Forensic and Security, and an Associate Editor for the IEEE Transactions on Circuits and Systems—Part I: Regular Papers and IEEE Transactions on Very Large Scale Integration (VLSI) Systems. He also served as the Associate Editor for the IEEE Transactions on Information Forensic and Security and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 2016 to 2019, IEEE Access from 2013 to 2019, IEEE Transactions on Circuits and Systems—Part I: Regular Papers from 2010 to 2013, *Integration, the VLSI Journal* from 2013 to 2015, *Journal of Hardware and System Security* (Springer) from 2016 to 2020, and *Microelectronics Journal* from 2014 to 2020. He guest edited several journal special issues and served on the organizing and technical program committees of more than 60 international conferences (mostly IEEE). He is an IET Fellow and was an IEEE Circuits and Systems Society Distinguished Lecturer from 2018 to 2019.

Weiqiang Liu (Senior Member, IEEE) received the B.Sc. degree in information engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2006, and the Ph.D. degree in electronic engineering from Queen's University Belfast, Belfast, U.K., in 2012.

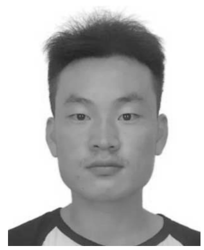
In December 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently a Professor and the Vice Dean. He has published one research book by Artech House and over 100 leading journal and conference papers. His research interests include approximate computing, hardware security, and VLSI design for digital signal processing and cryptography.

Dr. Liu has two Best Paper Candidates in IEEE ISCAS 2011 and ACM GLSVLSI 2015. He has been awarded the prestigious Outstanding Young Scholar Award by the National Natural Science Foundation of China in 2020. His paper was selected as the Feature Paper of IEEE TC in the 2017 December issue. He serves as the Associate Editor for IEEE Transactions on Circuits and System—Part I: Regular Papers from January 2020 to December 2021, IEEE Transactions on Emerging Topics in Computing from May 2019 to April 2021, and IEEE Transactions on Computers from May 2015 to April 2019, and a Steering Committee Member of the IEEE Transactions on Multi-Scale Computing Systems from January 2018 to December 2019. He is the Program Co-Chair of IEEE ARITH 2020, and also a Technical Program Committee Member for ARITH, DATE, ASAP, ISCAS, ASP-DAC, ISVLSI, GLSVLSI, SiPS, NANOARCH, AICAS, and ICONIP. He is a member of CASCOM and VSA Technical Committee of IEEE Circuits and Systems Society.



Shichao Yu received the B.S degree in electrical and information engineering from Hangzhou Normal University, Hangzhou, China, in 2014, and the M.S. degree in electronic circuit and system from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2017. He is currently pursuing the Ph.D. degree in electrical and electronic Engineering with Queen's University Belfast, Belfast, U.K.

His research interests mainly include secure hardware architecture, and hardware Trojan detection and its software/hardware implementation.



Yale Wang received the B.S. degree in automation from the Luoyang Institute of Science and Technology, Luoyang, China, in 2013, and the M.S. degree in information engineering from the Henan University of Science and Technology, Luoyang, in 2016. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His research interests mainly include PUFs and machine learning-based modeling attacks on PUFs and countermeasures.

Máire O'Neill (Senior Member, IEEE) received the Ph.D. degree from Queen's University Belfast, Belfast, U.K., in 2002.

She is a Regius Professor of Electronics and Computer Engineering with Queen's University Belfast, where she is the Director of the Institute of Electronics Communications and Information Technologies and the Centre for Secure Information Technologies. She is also the Director of the £5M EPSRC/NCSC-funded Research Institute in Secure Hardware and Embedded Systems and recently led the €3.8M EU H2020 Secure Architectures for Future Emerging Cryptography (SAFEcrypto) Project from 2014 to 2018. She previously held a U.K. EPSRC Leadership Fellowship from 2008 to 2014 and was a Former Holder of a U.K. Royal Academy of Engineering Research Fellowship from 2003 to 2008. She has authored two research books, and over 160 peer-reviewed international conference/journal publications. Her research interests include hardware cryptographic architectures, lightweight cryptography, side-channel analysis, PUFs, and post-quantum cryptography.

Dr. O'Neill has received numerous awards, which include the Blavatnik Engineering and Physical Sciences Medal in 2019, the Royal Academy of Engineering Silver Medal in 2014, and the British Female Inventor of the Year 2007. She is an Associate Editor of IEEE Transactions on Computers and IEEE Transactions on Emerging Topics in Computing and is the Secretary of the IEEE Circuits and Systems for Communications Technical committee. She is a member of the U.K. AI Council. She is a Fellow of the Royal Academy of Engineering and Irish Academy of Engineering and a member of the Royal Irish Academy.