Countering Modeling Attacks in PUF-based IoT Security Solutions

WASSILA LALOUANI, MOHAMED YOUNIS, MOHAMMAD EBRAHIMABADI, and NAGHMEH KARIMI, University of Maryland, Baltimore County

Hardware fingerprinting has emerged as a viable option for safeguarding IoT devices from cyberattacks. Such a fingerprint is used to not only authenticate the interconnected devices but also to derive cryptographic keys for ensuring data integrity and confidentiality. A **Physically Unclonable Function (PUF)** is deemed as an effective fingerprinting mechanism for resource-constrained IoT devices since it is simple to implement and imposes little overhead. A PUF design is realized based on the unintentional variations of microelectronics manufacturing processes. When queried with input bits (challenge), a PUF outputs a response that depends on such variations and this uniquely identifies the device. However, machine learning techniques constitute a threat where intercepted **challenge-response pairs (CRPs)** could be used to model the PUF and predict its output. This paper proposes an adversarial machine learning based methodology to counter such a threat. An effective label flipping approach is proposed where the attacker's model is poisoned by providing wrong CRPs. We employ an adaptive poisoning strategy that factors in potentially leaked information, i.e., the intercepted CRPs, and introduces randomness in the poisoning pattern to prevent exclusion of these wrong CRPs as outliers. The server and client use a lightweight procedure to coordinate and predict poisoned CRP exchanges. Specifically, we employ the same pseudo random number generator at communicating parties to ensure synchronization and consensus between them, and to vary the poisoning pattern over time. Our approach has been validated using datasets generated via a PUF implementation on an FPGA. The results have confirmed the effectiveness of our approach in defeating prominent PUF modeling attack techniques in the literature.

CCS Concepts: • **Security and privacy → Systems security**;

Additional Key Words and Phrases: Hardware fingerprinting, device authentication, replay attack, physically unclonable functions, IoT, security

ACM Reference format:
Wassila Lalouani, Mohamed Younis, Mohammad Ebrahimabadi, and Naghmeh Karimi. 2022. Countering Modeling Attacks in PUF-based IoT Security Solutions. *J. Emerg. Technol. Comput. Syst.* 18, 3, Article 46 (March 2022), 28 pages.
https://doi.org/10.1145/3491221

1 INTRODUCTION

The major technological advances in recent years have enabled the incorporation of computation and communication capabilities in all sorts of devices. The notion of **Internet of Things**

Authors' address: W. Lalouani, M. Younis, M. Ebrahimabadi, and N. Karimi, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland, 21250, USA; emails: {lwassil1, younis, ebrahimabadi, nkarimi}@umbc.edu.

1550-4832/2022/03-ART46 $15.00
https://doi.org/10.1145/3491221

(IoT) refers to internetworking these devices at a large scale to serve a broad range of application domains, such as healthcare, transportation, manufacturing, military, and tourism [1]. However, the limited resources, pervasiveness and unsupervised operation, makes IoT devices vulnerable to security attacks. Specifically, a device could be tampered with and manipulated. Moreover, the diversity in device vendors allows cloning and malicious usage of replicated devices. Therefore, IoT security is quite challenging and requires unconventional solutions.

As mentioned, the pervasiveness nature of IoT makes device authentication very crucial. Two categories of authentication methodologies have been pursued in the literature. The first relies on cryptography systems where digital signatures and keys are stored on the devices. Clearly, such a methodology is not robust since adversaries can apply side channel analysis attacks to the physical device to retrieve the key, or fabricate a replicated circuitry to impersonate the real device [13, 33]. The second methodology relies on unique device fingerprinting. Among the most notable techniques in this category are those employing **Physically Unclonable Functions (PUFs)** [4, 11, 21].

A PUF design exploits variations in the device fabrication process to generate a fingerprint. A PUF simply maps an input bit stream, referred to as challenge, to an output, referred to as response. Since the process variations are random in nature and are independent of the fabricated devices, a PUF cannot be cloned and its challenge-to-response mapping constitutes a signature for the device. A PUF is deemed as an attractive choice for supporting IoT authentication since neither the challenge nor the response is stored aboard the device [11, 13]. However, it has been shown that advanced machine learning techniques could successively model the PUF operation using some intercepted **challenge-response pairs (CRPs)** for training [10]. In fact, modeling the PUF via machine learning techniques has been easier given the awareness of the PUF design despite not knowing the process variation. The focus of this paper is on mitigating the vulnerability of PUF-based authentication schemes to modeling attacks. A simple and lightweight mechanism is proposed to diminish the risk of the PUF modeling and enable robust PUF-based authentication of embedded devices.

Our mechanism counters the threat of PUF modeling using **Adversarial Machine Learning (AML)** methodologies. AML opts to degrade the accuracy of data-driven models by injecting erroneous input. In the context of PUFs, such erroneous input corresponds to wrong association of challenge and response. Such a poisoning approach should not be predictable to an eavesdropper, yet both the server and client need to have agreement on the poisoning pattern. Moreover, AML techniques are demanding in terms of computational resources and would thus constitute a major burden if needed to be applied by an IoT device. To overcome these challenges, we propose a **Coordinated and Lightweight AML-based Countermeasure** for PUF modeling attacks **(CoLAC).** Our countermeasure strives to degrade PUF modeling attempts while minimizing the load on the IoT device and making it difficult for the adversary to distinguish between legitimate and erroneous CRPs. CoLAC achieves its design goal by making the following contributions:

- *Defense through implicit coordination:* While the complexity of AML will be unbearable for an IoT device, it is compatible with the capabilities of the server. AML involves two decisions, namely, when data poisoning is applied and what data to be poisoned in order to make the most impact on the adversary. CoLAC applies a simple strategy for the latter and instruments implicit coordination for the former, i.e., deciding when to poison. *Such implicit coordination also allows both legitimate and poisonous data to be used for authentication;* thereby no communication traffic overhead is imposed.
- *Approximate estimation of adversary's success:* By intercepting the exchanged CRPs, an adversary accumulates knowledge over time that translates to growing accuracy of the PUF

model. AML-based data poisoning opts to degrade such a model. CoLAC triggers such poisoning when the model accuracy reaches a level that is deemed to be a threat. Tracking the growth of the adversary's model accuracy would require mimicking the modeling process, which in essence implies running a machine learning algorithm for the used CRPs repeatedly. To avoid such computational burden, CoLAC uses the ratio of poisoned CRPs as an approximated measure for accuracy; this allows the device and server to easily reach a mutually agreed upon data poisoning decision without an explicit exchange of coordination messages.

- *Randomizing the data poisoning pattern:* The AML strategy would be defeated if the adversary could distinguish between legitimate and erroneous CRPs. Given how AML selects what data to poison, an adversary cannot differentiate between CRPs based on the values of challenge and response. Yet, the adversary can do that if a certain poisoning pattern could be inferred. CoLAC tackles that by introducing randomization in the poisoning decision process. A pseudo random number generator is employed at both the device and server that enables the generation of time-varying sequence of random numbers in a synchronized way. Such randomization makes it difficult for the adversary to predict when poisoning will take place while allowing both the device and server to implicitly agree on whether the next CRP is legitimate or erroneous; although we use both legitimate and erroneous CRPs for authentication, the corresponding responses are treated differently.

In summary, CoLAC is a lightweight mechanism that enables attack-resilient PUF-based authentication of IoT devices without requiring hardware modification. We validate the robustness of CoLAC using FPGA-generated dataset. The remainder of the paper is organized as follows. The next section discusses the related work. Section 3 goes over our system and attack models. Section 4 explores the ability of conventional adversarial machine learning techniques in countering PUF modeling attacks. The analysis of Section 4 will be used to devise our proposed simple and effective PUF modeling countermeasure in Section 5. Section 6 reports the validation results. Finally, the paper is concluded in Section 7.

2 RELATED WORK

As mentioned in the previous section, a PUF is deemed as an attractive security primitive. Particularly, strong PUFs are used for supporting device authentication. The focus of this paper is on countering ML modeling attacks against PUF-based security solutions. Given the scope, we cover related work on PUF-based authentication and published defense mechanisms against modeling attacks. We note though that some published work has just studied the use of PUF to generate cryptographic keys, without explicit consideration of device authentication [26, 34].

2.1 PUF-based Authentication

To support the secure transfer of data in IoT, the authors of [8] benefit from asymmetric cryptographic schemes. They deploy PUFs in order to generate public and private keys to be used for such data transfer. The proposed scheme is resilient against replay attacks, yet it is computationally intensive. Wallrabenstein et al. [36] include a hardware implementation of an Elliptic Curve cryptosystem on the IoT device to support secure transfer of data, and propose to embed a PUF along with this cryptosystem to regenerate the private key when needed. However, the approach requires some changes to be applied to the IoT hardware. Some work pursues multi-factor authentication, e.g., by using a shared cryptographic key in addition to the CRP [19]. The focus of Chatterjee et al. [9] is on avoiding the storage of CRPs on the server. They have proposed a solution that incorporates identity-based encryption and keyed hash functions along with the PUF, in the

authentication process. Mahalat et al. [27] have proposed a PUF-based mechanism for authenticating IoT devices that are interconnected through WiFi links. Aman et al. [3] have also developed a message-exchange protocol for authenticating devices using their PUFs.

Although the aforementioned authentication schemes are secure due to the unclonability nature of PUFs, they either impose high overhead or are vulnerable to PUF modeling attacks where the adversary can build the PUF model using a subset of the intercepted CRPs. Such modeling can eventually facilitate replay, impersonation and other IoT related attacks, and should thus be mitigated. CoLAC tackles such vulnerability. On the other hand, in [2] both device and communication link fingerprinting are pursued for authentication. The former is PUF-based, while the latter is through matching the channel characteristics of the wireless link between the IoT device and the verifier. A similar idea has been explored in [22]. Overall, the wireless channel fingerprinting is too sensitive to channel noise variation and could in fact hinder successful authentication of legitimate connection requests.

2.2 Modeling Attack Countermeasures

Employing machine learning techniques have been deemed effective in modeling the PUF and thus violating its randomness property [10, 32]. To counter such a threat, several methods have been proposed in the literature. Aman et al. [4] have conducted mutual authentication of IoT nodes with a sequence of dependent challenges via generating each challenge bit-stream based on the previous one. However, this method is vulnerable to impersonation if one challenge is leaked. Incorporating a fake PUF along with the legitimate PUF has been pursued in [20]. The fake PUF is queried intermittently so that wrong CRPs are introduced in the adversary's PUF modeling dataset. Obviously, this method suffers from hardware and communication overhead. In addition, only the challenge response pairs related to the legitimate PUF are used for authentication. CoLAC, on the other hand, uses both correct and poisoned responses to authenticate the device, and thus imposes no communication overhead. The approach of [40] is to limit the number of used CRPs in order to cap the attack accuracy. However, repeated usage of CRPs constitutes vulnerability for the device to message replay attacks; hence such an approach is not applicable if authentication is required very often. To mislead attackers, adversarial machine learning has been exploited by Wang et al. [37]. Unlike CoLAC, the poisoning of the response is based on the challenge bits. However, such a scheme has been defeated using Neural Networks, achieving PUF modeling accuracy of 96% [45]. The authors also studied sequential data poisoning; yet the poisoning pattern is regular and can be inferred. The scope of the work has been extended in [44], where the poisoning decision is made by a function that relies on both the challenge bits and a specific bit-string; the latter is generated either by an additional weak-PUF embedded in the circuit or via storing possible values on chip, and hence imposing hardware changes. A key advantage of our CoLAC mechanism is that it can be implemented without any hardware modification and hence is applicable to already-deployed devices. Ebrahimabadi et al. [45] also utilize AML to introduce noisy data that degrades PUF modeling attempts. Although successful against modeling attacks, the approach introduces high computational overhead due to conducting ML-based modeling quite frequently.

Some schemes have employed a cryptosystem in order to mitigate the PUF modeling vulnerability. For example, Vatajelu et al. [35] have proposed to encrypt challenge-bit streams via using a hardware implementation of the AES algorithm, where the encryption key is generated using a weak PUF. This method suffers from the large area overhead imposed due to the implementing AES as well as adding an extra PUF for key generation. The approach of Gope et al. [18] does not transmit responses; instead it uses the PUF output to generate a pseudo response through a sequence of steps that are known to the communicating parties. The server includes a random number (nonce) and employs a hashing function in its request; such a number is used by the device in generating

the pseudo response. Similarly, PUF-IPA [14] applies a cryptographic hash of the PUF response and stores only hashed (and encrypted) values in the database that are securely accessible by the server. Although the SRAM-PUF based authentication scheme of [30] uses the SRAM address instead of the challenge, it still applies a cryptographic hash and uses a nonce. Overall, this category of schemes simply loses the PUF advantage by employing a cryptographic hash function which constitutes significant computational overhead for the devices. CoLAC avoids such overhead. Also, the hashing function needs to be agreed upon by the communicating parties. In addition, repeating the nonce makes the system vulnerable to message reply and man-in-the-middle attacks.

Some techniques pursue a hardware-based methodology for countering modeling attacks by either changing the PUF design or augmenting the PUF with additional circuits. For example, Ganji et al. [17] takes advantage of programmable logic, e.g., FPGA, and proposes rolling out some of the PUF stages, i.e., swapping them. To maximize the impact of the rollout, the most influential stages (challenge bits) on the ML-model accuracy are picked. Meanwhile, PHEMAP [5] uses a sequencer where the challenge C_i at time t_i is a function of $C_0, C_1, \ldots, C_{i-1}$. The interpose PUF [29] pursues a variant of the XOR PUF; yet it has been recently shown to be vulnerable to modeling attacks [38]. Dubrova et al. [12] use a CRC circuit to shuffle the challenge bits so that the adversary fails to correlate the response to the challenge. The CRC polynomial is deemed to be a secret that the device and server agree on. Gassend et al. [15] use a "Controlled" PUF that mutates the challenge and response with hash functions.

On the other hand, Ma et al. [25] uses a weak-PUF to obfuscate the challenge of the main strong PUF. Zalivaka et al. [42] have proposed an "Obfuscated" PUF design where the challenge is mutated before feeding the PUF; the mutation is based on a predefined algorithm realized in hardware. However, the incorporation of an additional circuit constitutes a major overhead and diminishes the advantages of PUF as an authentication primitive for resource constrained IoT devices. Yu et al. [41] embed PRNG in the XOR PUF design. The PRNG provides half the challenge bits and the other half comes from the server. An eavesdropper will not know all the challenges. In addition to the hardware overhead, the resilience of such an approach is questionable for an Arbiter-PUF as an attacker might model the PUF with a reasonable accuracy using a subset of the challenge bits since some challenge bits may have little impact on the PUF modeling accuracy Another approach that pursues challenge obfuscation is proposed by Majzoobi et al. [28], where the challenge bit stream is partially formed by the server and completed by the IoT device. The approach requires hardware support including the implementation of encryption and random number generation. The same argument applies to PUF-RAKE [31], where both the challenge and response are obfuscated through random shuffling.

In contrast to the aforementioned techniques, CoLAC is very lightweight, does not constrain the usage of the available CRP combinations, and can be fully implemented in software. Only the main PUF is implemented in hardware. A key advantage of CoLAC is its ability to ensure robust authentication while preserving the fundamental PUF properties, namely, randomness and uniqueness. Other advantages of CoLAC include: (1) it is not dependent on the PUF design and works with any strong PUF circuit, (2) it is adaptive to the security threat as it applies poisoning when the perceived accuracy of the machine learning model exceeds a certain bound, (3) it can be applied to legacy PUF-equipped devices, and (4) it imposes no hardware overhead.

3 SYSTEM AND ATTACK MODELS

3.1 System Model and Preliminaries

In practice, two similar semiconductor devices typically have slight differences due to the unintentional variations (imperfection) in the manufacturing process [21]. Such imperfection cannot be

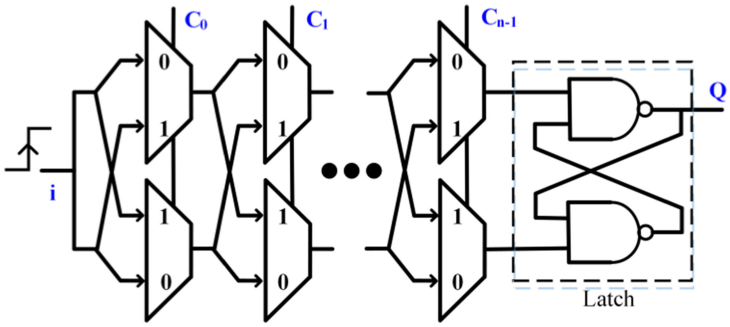


Fig. 1. Schematic diagram of an Arbiter-PUF, where the challenge bits control the individual multiplexers and cause the input signal to experience different delays on distinct devices and consequently the latched value (Q) would differ.

controlled by the manufacturer. The PUF design exploits such manufacturing variation to distinguish devices. In fact, even two similar PUF circuits implemented on the same silicon die behave differently. Figure 1 shows an example PUF design, known as Arbiter-PUF, where each of the signals $C_0, \ldots, C_{n-1}$ affects the setting of the corresponding multiplexer and consequently the path that the input signal travels until reaching the latch Q. Variations among devices will cause the input signal to experience different propagation delays and hence the latched value could vary. The bits $C_0, \ldots, C_{n-1}$ are referred to as the PUF challenge and the corresponding Q is called the PUF response. The size of PUF reflects the number of challenge bits, i.e., n. Querying the circuit in Figure 1 m times (with different challenges) enables the device to have a multi-bit PUF-based identifier that is unique per device and constitutes a fingerprint. In other words, two PUF circuits will provide different responses for a set of similar challenges. The strength of a PUF is characterized by the number of distinct CRPs it offers. The Arbiter-PUF, shown in Figure 1, is considered as one of the strong PUFs and will be used as an example in the balance of this paper; nonetheless, CoLAC can be applied to all other strong PUF designs. Note that week PUFs are mainly used for key generation rather than authentication.

Implementing PUFs is simple and does not impose much area overhead. Therefore, PUFs have been leveraged in devising hardware-assisted security solutions, especially for IoT where the involved devices are resource-constrained. Particularly, authenticating a device is conducted by checking the response of the embedded PUF to certain challenge bit-stream. The typical architecture, which we also assume, is to engage a trusted and secure server in the IoT. Such a server will be loaded with a table of a subset of the challenge-response pairs of the embedded PUF on the device. The server will send an authentication request to the device with a set of challenge bits, i.e., one of the entries in the table, and validate the device response against the table. The authentication process could be repeated periodically or based on application level criteria. Communication between the IoT device and the server is over wireless links, and can thus be intercepted as we explain next. We assume that reliable delivery of packets between the server and IoT devices is ensured, e.g., through acknowledgements. We also assume that the noise effect on the PUF output is mitigated through any of the conventional schemes such as the incorporation of error correction codes [24], or circuit-level majority voting over repeated challenge application [23].

3.2 Attack Model

The objective of the adversary is to crack the authentication process in order to impersonate devices, or inject malicious devices. The adversary will not have access to the embedded circuits

on an IoT device for reverse engineering and will instead eavesdrop on the communication link between the server and device, and then use the intercepted CRPs to model the PUF operation. Machine learning algorithms have been shown to be quite effective in PUF modeling [42]. In fact, by factoring in the PUF design methodology, machine learning could achieve high accuracy using a relatively small training dataset, i.e., CRPs [37]. By modeling the PUF the adversary would be able to clone the IoT device and could launch numerous subsequent attacks without being detected. Moreover, the adversary would be able to impersonate the legitimate device and conduct malicious activities that lead to expelling such a device from the network. Hence, PUF modeling attacks are real threats that could diminish the utility of the PUF-based fingerprinting. The existence of such adversarial behavior motivates the need to increase the resilience of the authentication mechanism against PUF modeling techniques. We note that an adversary does not have access to (capture of) the IoT device itself; hence attack strategies that exploit access to the PUF, e.g., reliability based attacks, e.g., [7], are not within scope. An example scenario is when the PUF is used to authenticate a node in a connected autonomous vehicles application, where accessing the vehicle's electronic system is not feasible.

In this paper, we mitigate machine learning based PUF modeling attacks by applying adversarial classification techniques, i.e., AML. AML strives to degrade the model accuracy by injecting erroneous data. Conventionally, AML is deemed a threat to data-driven designs; yet we use it as a defense mechanism against PUF modeling attacks. However, in the context of PUF-based authentication, an AML-based defense strategy ought to overcome the following challenges:

(1) It should prevent the adversary from distinguishing between correct and erroneous data, and adapting the attack accordingly, e.g., by filtering out erroneous data. Specifically, there should be no detectable pattern for the data poisoning process or discriminator indicator of the legitimate/illegitimate data.
(2) Given the resource-constrained IoT devices, the decision process for what erroneous data to inject and when to inject it should be lightweight in terms of processing and storage requirements. Most existing AML techniques assume the availability of large amounts of labeled instances. In addition, they have to consider the knowledge of the attacker when injecting the poisoning data which involves non-trivial optimization formulation and many constraints. A simple, yet effective, AML technique is needed.
(3) Both the server and the device have to coordinate in order to avoid failing the authentication process due to the poisoned data that are shared to counter the PUF modeling attacks.

In the next section we study the effectiveness of conventional techniques in countering the PUF modeling threat and then use such a study in devising our approach to meet the aforementioned requirements.

4 APPLYING CONVENTIONAL AML STRATEGIES

Adversarial machine learning constitutes a threat for data-driven models. CoLAC employs AML as a defense mechanism to thwart PUF modeling attempts. Given the nature of a data-driven methodology, the idea of AML is to inject erroneous data that affects the modeling accuracy and eventually leads to misclassification. Such a process is referred to as poisoning and evasion depending on whether the erroneous data is injected at the time of training or testing. In the context of PUF, we will use poisoning in both cases since we do not know the stage the adversary is at and how the intercepted CRP will be used. We have studied the effectiveness of popular AML techniques for countering PUF-modeling. In this section, we provide insight on how the PUF could be effectively modeled, and an overview of how AML could be applied. We further discuss the results of applying

some contemporary AML techniques. In the next section, we leverage the analysis and results of this section in designing CoLAC.

4.1 PUF Modeling

Assume that the adversary is building a classifier that can be parametrized by a set of weights W, determined using a dataset $S = \{CRP_i\}_{i=\gamma}^{\mu}$, and validated against the set $V = \{CRP_j\}_{j=\alpha}^{\beta}$. Here γ, μ, α, β are symbolic representations to show the size of training and test sets used in machine learning. The V and S sets are disjoint and their size can be similar or different. The classifier is fed with CRPs. We use the Arbiter-PUF as an example to illustrate how the modeling attack could be applied. Considering the architecture of an Arbiter-PUF shown in Figure 1, the propagation delay of the input transition to the output is dependent on the challenge bits and the delays of the multiplexers. Hence, the behavior of such a PUF circuit can be modeled by an additive delay function where the response bit is generated based on the sum of delays in all stages (each stage includes two multiplexers fed by the same challenge bit), where the delay of a stage depends on the corresponding challenge bit. The response for a challenge $C = (c_0, \ldots, c_{n-1})$ is determined based of the sign of the delay differences in the top and bottom paths feeding the arbiter (latch). Modeling an Arbiter-PUF can thus be captured by the following, often referred to as the mapping function [32]:

$$\begin{gathered}\Delta = \omega,\ \varphi \\ \omega = \left(\omega^0, \omega^1, \ldots, \omega^n\right) \\ \varphi(C) = \left(\varphi^0(C), \varphi^1(C), \ldots . \varphi^{n-1}(C), 1\right) \\ \text{where } \varphi^j(C) = \prod_{i=j}^{n-1}(1 - 2c_i)\ \ j = 0,\ 1,\ 2,\ 3, \ldots,\ n-1 \end{gathered} \tag{1}$$

where Δ is the sign of the difference of the propagation delays of two paths feeding the arbiter (S-R latch in Figure 1) ω is a vector of weights that reflect the influence of the multiplexers' delays, and φ is a function of input challenge bit-stream. Since the PUF operates in a cascaded way, the output of a stage depends on the challenge bit of such a stage and the delay of the previous stage. The mapping function captures such dependency by $\prod_{i=j}^{n-1}(1 - 2c_i)\ j = 0,\ 1,\ 2,\ 3, \ldots,\ n-1$. Assume ρ_i^0 and ρ_i^1 are the delays at stage i for the uncrossed and crossed signal paths, respectively. The objective is to determine the vector ω that encodes the multiplexer delays and use it along with the challenge bits to determine the response according to the sign of Δ using Equation (1). Here $\omega = (\omega^0, \omega^2, \ldots, \omega^n)$, where ω^i is calculated as follows [32]: $\omega^0 = \frac{\rho_1^0 - \rho_1^1}{2}$, $\omega^i = \frac{\rho_{i-1}^0 + \rho_{i-1}^1 - \rho_i^1 + \rho_i^0}{2}\ for\ i = 1,\ 2, \ldots, n-1$ and $\omega^n = \frac{\rho_{n-1}^0 + \rho_{n-1}^1}{2}$. Thus, the role of machine learning is to find the response by training the model to regulate ω.

When factoring in the mapping function, a **neural network (NN)** classifier yields 90% accuracy for modeling a 64-bit Arbiter-PUF with as low as 500 CRPs. This shows how serious the PUF modeling attack could be and why devising an effective countermeasure is important. Since the PUF modeling attack is in essence data-driven, we will use adversarial machine learning as a defense mechanism.

4.2 Countering PUF-modeling Through AML

Given the PUF modeling attack discussed above, AML opts to introduce a poisonous set of data, $P = \{CRP_k\}_{k=\delta}^{\gamma}$, to maximize a loss function $\mathrm{l}(W,\ CRP_j)$ evaluated over V, i.e., $j = \ \alpha,\ \alpha\_1,\ \ldots, \beta$. Such a loss function is relative to the accuracy of the PUF model. AML strategies can be divided according to their data manipulation, i.e., determining the set P, into gradient-based poisoning and

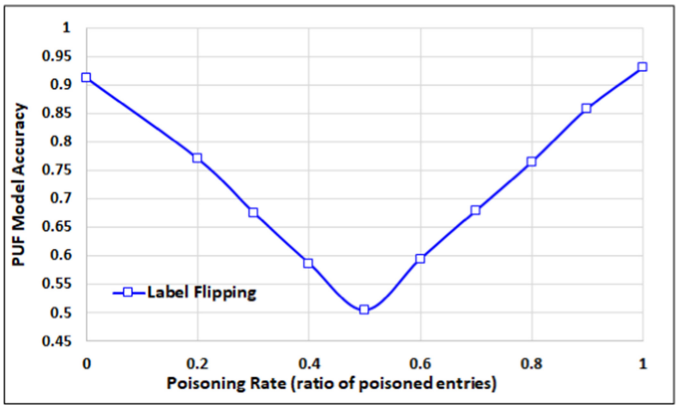


Fig. 2. The effectiveness of the label flipping adversarial machine learning strategy as a function of the poisoning rate.

label flipping [39]. The idea for both categories is to fool the PUF modeling classifier of the attacker by providing wrong CRPs; yet the approach is different. The gradient-based strategy strives to introduce noise to some of the data items such that they get misclassified. Determining the altered data is through formulating a complex optimization to maximize the loss in the model accuracy while limiting the noise amount to a minimum value, e.g., ε. Moreover, the results of the optimization is real numbers and optimality could be lost when approximated to binary values. Therefore, gradient based poisoning is not suited in countering PUF modeling attacks. Label flipping, on the other hand, strives to modify the attributes (label) of some data items such that their classification changes, i.e., flips from one class to another. Picking the set of data items to be flipped is also subject to optimization where the fewest items are selected to maximize the model loss. Nonetheless, the optimization formulation is significantly less complex compared to the gradient-based strategy. Moreover, label flipping can be applied without such an optimization. In other words, the data items could be picked randomly, and thus becomes more suited for applications where heavy computational overhead is undesirable. In the context of PUF modeling, label flipping simply modifies the response sent back from the IoT device. Regardless of the AML strategy, both the server and IoT devices have to independently determine when to poison and for what challenge. Given the advantages of label-flipping in terms of simplicity and being applicable using randomly picked challenges, it is deemed suitable for CoLAC. We have evaluated the effectiveness of such AML strategy using data generated from a 64-bit implementation of an Arbiter-PUF on a Xilinx ARTIX7 FPGA. For that we have followed the recommended steps in [29], where each stage of the Arbiter-PUF (composed of two multiplexers) is implemented using a 5-to-2 LUT. In the considered FPGA, each slice includes four LUTs, so-called BELs. Based on [29] suggestion, we used the same Bell to place the LUTs in each slice. We have then studied the performance of the label flipping scheme when 4,000 CRPs are exchanged (intercepted by the attacker). We assume that two-thirds of the CRPs are used for training and the remaining for testing. We vary the number of poisoned CRPs (poisoning rate) and monitor the accuracy during the test to assess the ability of the AML technique. The results are reported in Figure 2 and demonstrate the effectiveness of the label flipping strategy, where the accuracy could be diminished to about 50% which is ideal for binary classification like the case for PUF modeling. As the percentage of poisoned data increases, the accuracy rises again which is expected since the adversary will be able to predict the erroneous (poisoned) response; in other words, the adversary will be able to model the altered PUF behavior. In fact, the accuracy decline and rise, after the turning point, are proportional to the ratio of

poisoned CRPs in the dataset. These observations are very important since in CoLAC, coordination between the server and the IoT device is implicit and thus both the server and the device ought to apply AML to reach the same decision. In the next section we present the details of CoLAC.

5 PUF-MODELING COUNTERMEASURE

CoLAC opts to degrade the adversary's ability for modeling the employed PUF while: (i) avoiding overburdening the IoT device, and (ii) preventing the adversary from inferring the protection methodology. To achieve the design objectives, CoLAC: (1) employs AML to inject erroneous (poisonous) data to the adversary's model and diminishes its accuracy, (2) limits the overhead on the IoT by applying lightweight procedure for deciding when to transmit poisonous data, (3) instruments implicit coordination between the server and the IoT device so that no information is leaked about the protection scheme through message exchange, and (4) randomizes the data poisoning pattern so that the adversary cannot predict and filter erroneous data to defeat the provisioned protection. CoLAC can be viewed to consist of three functional modules, namely, coordinated AML application, poisoning pattern control, and poisoned data generation, as will be explained in the balance of this section.

5.1 Coordinated AML Application

As shown in the previous section, AML is an effective methodology for degrading the adversary's modeling attack on the embedded PUF. CoLAC employs AML, yet it avoids engaging the IoT device in complex computation. Fundamentally, applying AML is associated with two questions: (i) when data poisoning is warranted, and (ii) how the poisoned data is determined. The two questions obviously involve both the server and IoT device. Although data poisoning targets the adversary, if uncoordinated it would cause failure in authenticating the IoT device. In other words, both the server and the IoT device ought to mutually agree on whether a poisonous CRP will be generated next. Such coordination could be realized explicitly or implicitly. The former entails message exchange between the server and the IoT device; basically the server could be making the decision and just instruct the IoT device to send the wrong response to the challenge. However, such a strategy is vulnerable since the adversary could intercept these messages and differentiate between legitimate and poisonous responses. Recall that the adversary is assumed to eavesdrop on all message exchange between the server and the IoT device. Therefore, CoLAC favors implicit coordination where both communicating parties simultaneously and independently reach the same decision on whether to poison the next CRP or not.

In CoLAC, the decision to poison a CRP is based on the accumulated adversary's knowledge about the PUF. Basically, poisoning is warranted when the adversary intercepts a sufficient number of legitimate CRPs to yield an accurate PUF model. CoLAC provides a parameter for the maximum accuracy level that could be tolerated; such a level is PUF- and application-dependent. Note that poisoning all CRPs is not beneficial as the adversary can easily model the PUF in that case as well. The idea behind CoLAC is that when the adversary's model reaches such an accuracy level, the model should be degraded by injecting erroneous data. To assess the accuracy, one would have to run the same machine learning technique that the adversary applies. Firstly, such a technique is unknown to both server and the IoT node, although powerful modeling such as **Neural Networks (NN)** and **Support Vector Machine (SVM)** scheme would give enough insight on the accuracy that the adversary can achieve [32]. Secondly, although applying the ML technique is conceivable at the server, this will impose significant overhead on the IoT device. Therefore, CoLAC employs a simplified method by monitoring the running average of the legitimate CRPs over time; we refer to such an average as the **perceived accuracy (PACC)**, which constitutes a measure of information leakage about CoLAC's operation. The idea is based on the conclusion from Section 4 (shown as

Label Flipping in Figure 2), where the degradation in PUF modeling accuracy is proportional to the percentage of poisonous data. Thus, by keeping the ratio of legitimate CRPs to the total number of exchanged CRPs below some specific threshold, the accuracy of the adversary's model could be capped. On the other hand, given that poisoning in the context of PUF is a simple negation of the response, excessive poisoning may allow an adversary to model the PUF behavior. Assuming P and N to be the number of poisonous and legitimate (not poisoned) CRPs, we define the poisoning rate as:

$$PRate = \frac{P}{P+N} \tag{2}$$

As a rule of thumb, the perceived PUF modeling accuracy using Q and $\bar{Q}$ are:

$$PAcc_N = (1 - PRate) = \frac{N}{P+N} \tag{3}$$

$$PAcc_P = PRate = \frac{P}{P+N} \tag{4}$$

The perceived accuracy of the PUF model will thus be:

$$Perceived\ Accuracy\ \ (PACC) = \max(PAcc_N,\ PAcc_P) = max\left(\frac{N}{P+N}, \frac{P}{P+N}\right) \tag{5}$$

CoLAC opts to cap the model perceived accuracy, e.g., Equation (5). Since $PAcc_P = 1 - PAcc_N$, CoLAC monitors only $PAcc_N$ and employs two bounds. The upper bound corresponds to when legitimate CRPs dominates, and the lower bound opts to prevent PUF modeling through the poisoned CRPs. We will refer to these bounds by U_B and L_B, respectively. To avoid exceeding U_B, P should grow so that $PAcc_N$ diminishes. Meanwhile, $PAcc_P$ reflects the perceived modeling accuracy based on using the poisonous data; thus, L_B implies sustaining the value of N relatively high, and consequently P is to decrease. Since the poisoning decision depends on the poisoning rate that can be simultaneously monitored by the IoT device and server, both communicating parties will be synchronized. Both the server and IoT consider the perceived accuracy of the adversary's PUF model. However, as will be confirmed by the experiments in Section 5, the poisoning pattern should also remain unpredictable. CoLAC provides a provision to inject randomness in the poisoning pattern as discussed in the next subsection.

To address the second question about how to determine the data to be poisoned, we note that the attacker uses the intercepted CRPs to model the PUF. Thus, a poisoned CRP simply associates the wrong response to the challenge. Given the binary nature of the response, poisoning the response is simply through toggling the output of the PUF circuit. Thus, the IoT device role in the poisoning process is very lightweight and it needs to just know when to change the response. Through implicit coordination, CoLAC makes it easy for the IoT device to determine whether the response needs to be complemented (negated) and thus CoLAC completely spares the IoT device from any complex computation. The server, on the other hand, could apply more elaborate AML to identify the best challenge to use so that the adversary model is degraded the most; however, we found such complexity to be unwarranted, as will be shown in Section 6. In other words, CoLAC is lightweight for the server as well and thus enables scalability for large IoT networks.

Note that as mentioned in Section 3.1, we assume that the noise effect on the PUF output is mitigated through the incorporation of error correction codes [24], or circuit-level majority voting over repeated challenge application [23]. With such noise mitigation, the server receives very few noisy responses (in the case of pursuing repeated challenge applications) or finds out that the received response is not what expected (when error correction codes that are used and the corrected part does not match the data). In both cases the server updates its running average to be

in sync with the node but may query the node again with another challenge to find out whether the mismatch was due to noise or due to a malicious attack, e.g., impersonation.

5.2 Poisoning Pattern Control

One of the main advantages of CoLAC is the implicit coordination between the server and the IoT device. Such coordination not only avoids vulnerabilities caused by explicit message exchange, but also allows the poisoned CRPs to be in fact a means for authentication. If the server concludes that poisoning is to take place and the IoT device follows the prescribed process by providing the negated version of the correct response, the server considers that as an attestation of the authenticity of the IoT device. In the previous subsection, we have alluded to how the two communicating parties arrive at the same conclusion by monitoring the percentage of legitimate CRPs over time. We note, nonetheless, that an adversary may apply techniques like **recurrent neural networks (RNN)** to infer the poisoning pattern and filter the erroneous data. Particularly, **Long-Short Term Memory (LSTM)** models are well suited for such analysis. In Section 6, we show how LSTM may be applied for such a purpose. Thus, in order to sustain the effectiveness of the employed AML-based strategy, CoLAC introduces randomization in the poisoning pattern.

CoLAC introduces consensus-based randomization in the poisoning decision process. Both the IoT device and the server generate a time-varying sequence of random numbers in a synchronized way using the same **pseudo random number generator (PRNG)**. Such randomization makes it difficult for the adversary to predict when poisoning will take place while allowing both the device and the server to implicitly agree on whether the next CRP is legitimate or erroneous. Both the IoT device and the server need to apply a pre-agreed upon PRNG algorithm and seed. *We note that IoT devices may employ different PRNGs, as long as each pre-agrees with the server*, thus any gained knowledge about a certain device, cannot in general be applicable to another. The seed also can be derived from an **identification number (ID)** of the device. At the device side, the PRNG and seed generation functions could be implemented in hardware or software. At the time of device enrollment, which is often controlled and assumed to be secure, the server gets to know the ID of the IoT device and the employed PRNG algorithm and seed generation function based on the device ID. The specific PRNG and seed generation function will be applied by the server in software. We note that the strength of CoLAC is not solely dependent on the secrecy of the PRNG and the seed. As we discussed above, CoLAC fundamentally poisons the CRPs collected by the eavesdropper, where the poisoning process is adaptively activated based on the perceived accuracy of the ML modeling attack; the generated random numbers are just used to make the poisoning pattern unpredictable, as we explain below and validate in Section 6.

An important issue is the uniformity of the distribution of generated random numbers. Some generators could naturally yield non-uniform distributions; moreover, the system designer could just decide to introduce non-uniformity by combining multiple popular seed-based approaches so that it becomes much harder for an adversary to infer the distribution [16]. However, non-uniformity of the generated random numbers could let the accuracy of the attacker model intermittently exceed the desired level and risk the entire authentication process. Therefore, CoLAC does not make any assumption on which random number generation methodology is used and on the distribution of the generated random numbers, e.g., whether uniform or not. Basically, CoLAC pursues a quite general approach that controls the poisoning process so that the accuracy of the adversary's model is capped. In Section 6, we study the effect of the uniformity of the random number distribution on the performance of CoLAC.

Since both the IoT device and the server use the same PRNG, and apply the same poisoning decision process, the PRNGs at both communicating parties will be synchronized and generate similar sequences of random numbers. The process goes as follows. The server considers the perceived

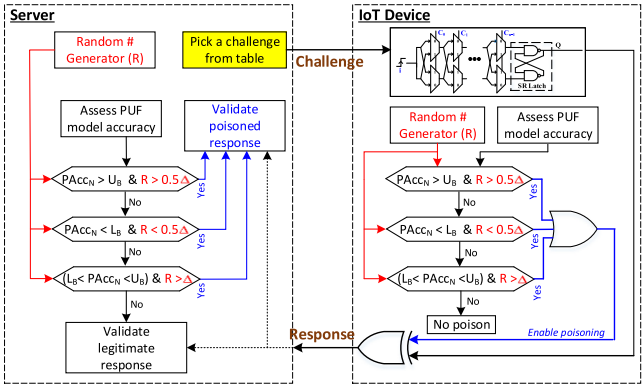


Fig. 3. Block diagram summary of the operation of CoLAC at both the server side and at the IoT device.

accuracy of the adversary's PUF model; for that a running average of legitimate CRPs is used, i.e., $PAcc_N$. When $PAcc_N$ surpasses a certain bound, e.g., 60%, poisoning at a certain rate is deemed necessary to cap the accuracy. To avoid periodic poisoning patterns, a random number, $R \in [0, 1]$, is generated and compared with a qualifier, $\Delta \in [0, 1]$. When $PAcc_N$ surpasses the considered upper bound and $R > 0.5\Delta$, the CRP is poisoned. In practice, the qualifier is meant to introduce variability for when poisoned CRPs are exchanged after the accuracy bound is violated. For example, if R follows a uniform random distribution, setting $\Delta = 0.5$ implies 75% probability of poisoning a CRP. The same value of Δ will be used at the server and IoT device.The value of Δ is dependent on the distribution of the implemented PRNG and is determined during the node enrollment in IoT. In the next subsection, we discuss the effect of Δ on the performance.

Again, the PRNGs at both communicating parties are synchronized and will yield the same number which is to be compared with the same Δ. Recall also that CoLAC assumes guaranteed message delivery through the underlying communication protocols, e.g., at the link or transport layers. Thus, the server and IoT device will have the same statistics about how many poisoned CRPs have been exchanged, and both will calculate the same value of $PAcc_N$ and come to the same conclusion. The entire CoLAC scheme is summarized in Figure 3. $PAcc_N$ will be updated after every exchanged CPR, both legitimate and poisoned. AML is applied all the time at a rate consistent with $PAcc_N$. CoLAC strives to keep $PAcc_N$ within an interval that prevents the adversary from modeling the PUF using legitimate or poisonous CRPs. When such an accuracy estimate is below a preset lower bound, L_B, AML is deemed as unwarranted in order to prevent modeling using poisoned CRPs; yet CoLAC poisons at a very low, rather than zero, rate in order to sustain the randomness of the poisoning pattern. Similarly, when $PAcc_N$ exceeds an upper bound U_B, CoLAC boosts the poisoning rate to decrease the accuracy. When $L_B < PAcc_N < U_B$, CoLAC strives to sustain such accuracy by balancing the frequency of poisoned and legitimate CRPs.

Injecting randomness in the poisoning patterns is by employing a PRNG. To determine whether a poisonous CRP is pursued next, the generator is queried to get a random number and such a number is compared to a qualifier, Δ. The use of such a qualifier is based on the relationship between $PAcc_N$ and both L_B and U_B. When $L_B < PAcc_N < U_B$, poisoning will take place only if the random number exceeds Δ. Since CoLAC opts to keep $PAcc_N$ within the bounds, the value of Δ should enable an approximately 50% poisoning rate so that the accuracy does not change much. When $PAcc_N$ is out of bounds, the qualifier is adjusted to diminish or boost the poisoning rate based on whether L_B or U_B is violated, respectively. In Figure 3, the value of Δ is adjusted by a factor of ½ to boost the poisoning rate, i.e., having higher probability for the random number to

exceed the qualifier value. To decrease the poisoning rate, Δ is still adjusted by a factor of ½ while the comparison is negated. We note that the ½ factor is just an example to illustrate the idea; the adjustment of the qualifier will be discussed in the next subsection.

As explained in Section 4, AML-based label flipping is to pick the challenge bit pattern for which a poisoned response causes the most degradation of the adversary's PUF model. Such selection involves solving an optimization formulation. Nonetheless, CoLAC takes advantage of the fact that label flipping can be applied through random selection of the misclassified data items, i.e., challenges. Thus, regardless of whether a poisonous or legitimate CRP is pursued, CoLAC selects the challenge based on whatever criteria the server applies, e.g., sequentially from a table of device specific CRPs. As shown in Section 6, the performance will not be affected much compared to selection of challenges using an elaborate label flipping optimization.

5.3 Poisoned Data Generation

In order to ensure the robustness of our PUF modeling countermeasure, CoLAC pursues a probabilistic poisoning pattern while balancing between the poisoning/legitimate CRP rates. The goal is not only to degrade the PUF modeling ability of the adversary but also to keep the pattern of the poisoned CRP obscured. CoLAC strives to maintain $PAcc_N$ within the specified range regardless of the random number distribution. Figure 4(a) shows a state diagram representation of CoLAC operation. While Figure 3 captures the interaction between the server and IoT device, Figure 4(a) tracks the accuracy and focuses more on the poisoning data generation, specifically when the response of a challenge is to be inverted. The accuracy of the model is captured by $PAcc_N$, and is represented by three states, Low, Medium and High. Again $PAcc_N$ reflects what the IoT device and the server perceive about the adversary's model accuracy. The Medium state is deemed the ideal in terms of protection since $PAcc_N$ is within the desired range, i.e., $L_B \leq PAcc_N \leq U_B$. The High state reflects a major increase in $PAcc_N$ that warrants growing the poisoning rate; the Low state corresponds to excessive poisoning that could risk modeling the PUF. Being in any of these three states is based on the value of $PAcc_N$; meanwhile, the transition to poisoning and legit states depends on the random number (RND).

As shown in Figure 3, the decision on whether to poison the next CRP would depend on the value of the random number relative to Δ. If the generated random numbers are uniformly distributed in the range [0, 1], the poisoning rate will be very much regulated and the accuracy of the adversary's model will stay close to the desired pound. However, non-uniformly distributed random numbers could lead to a scenario where a long sequence of CPR exchanges go without poisoning and allow the adversary's modeling accuracy to grow to an undesirable level. Although the selection of what PRNG to employ is a system designer decision where certain PRNG properties can indeed be ensured, CoLAC avoids dependence on the uniformity profile of the generated random numbers. Such a feature keeps CoLAC effective regardless of which PRNG is being picked. We validate the effect of the uniformity of the PRNGs in Section 6.

If the distribution of the generated random numbers is known, the median would be the appropriate setting of Δ. If the PRNG is uniform, making $\Delta = 0.5$ will be ideal as the number of legitimate and poisoned CRPs will be balanced and $PAcc_N$ will stay in the Medium state. However, as noted above, CoLAC does not make assumptions about the PRNG. Figure 3 illustrates the overall coordination mechanism where the poisoning rate is regulated by a factor of 0.5, i.e., diminished or boosted by 50% when the accuracy violates L_B and U_B, respectively. In the state diagram of Figure 4(a) we are generalizing such a regulator, denoted by α, to be used along with the qualifier Δ. The setting of α depends on the state of the modeling accuracy and is pre-agreed upon between the server and IoT device. The objective is to instrument fine-grained control of the poisoning pattern. In the Medium state, the value of α is set to 1 implying that Δ will control the poisoning

rate. However, if the generated random numbers are not uniformly distributed and the median is unknown, such a balance could be violated and $PAcc_N$ will be in the Low or High state depending on the skewness of the random numbers distribution. The regulator α is responsible for adjusting the poisoning rate in such a case. Basically, being in the High state, implies insufficient poisoning because the random numbers distribution is skewed towards the range $[0, \Delta]$, which causes the transmission of legitimate CRPs to be more frequent than poisoned CRPs. Thus, decreasing α opts to adjust the poisoning rate. Similar logic applies when being in the Low state. Note that poisoning takes place in the High state when RND $> \alpha.\Delta$, and in the Low state when RND $< \alpha.\Delta$; thus diminishing the value of α would suffice in both cases.

Now we would like to direct attention to how to set α and Δ, and the implication of their settings on the performance of CoLAC. As also indicated by the results in Figure 2, ideally $PAcc_N$ should be around 50%, which means that for the attacker it is equally probable for a PUF response to be legitimate or poisonous. In essence, U_B and L_B reflect how much deviation from 50% $PAcc_N$ could be. Hence, it is often the case that (U_B − 0.5) is equal to (0.5 - L_B) and consequently the same setting of α could be used in the Low and High states. Generally, the setting of α is subject to tradeoff. On the one hand, a small α quickly adjusts $PAcc_N$, and thus accelerates the return to the Medium state. On the other hand, it is desirable to switch between transmissions of poisoned and legitimate CRPs in order to obfuscate the poisoning pattern and thus having $\alpha = 0$ would not be preferred. The best value of α is dependent on the skewness of the random numbers distribution. An option is to set α to a small constant, e.g., 0.3, to accelerate the return of $PAcc_N$ to within the desired range. In Section 6, we experiment with a static setting of α. We also check the effect of having $\alpha = 0$, which means that the High and Low states imply always poisoning and not poisoning, respectively. Such a case is marked in dotted brown arrows in Figure 4(a). Finally, α could be gradually diminished overtime to expedite the return to the Medium state. If such a dynamic change is implemented, it should be pre-agreed upon between the server and IoT device. For example, in the High state α could start with a value of 0.7 and decrease by a factor, e.g., 2, for every number, say 5, of consecutive legitimate CRPs, or diminish by a factor that is proportional to how far $PAcc_N$ is from U_B. A comparison of the static and dynamic setting of α will be provided in Section 6. The following Lemmas prove key CoLAC properties.

Lemma #1. *CoLAC guarantees staying within bounds with dynamic or zero setting of α in the Low and High states.*

Proof. We note that ideally RND should be compared with the median of the random number distribution. Nonetheless, CoLAC applies an adaptive poisoning rate adjustment strategy that is independent of the employed PRNG. Let's first consider the case when $\alpha = 0$ in the Low and High states. In such a case, the state diagram is reduced to what is shown in Figure 4(b), where the High and Low states become equivalent to the Poison and Legit states, respectively. In essence, the randomness of the poisoning pattern is ignored when the accuracy range is violated. Based on Equation (3), $PAcc_N$ is inversely proportional to P; thus, by continuously poisoning, the $PAcc_N$ will diminish and eventually return to the Medium (safe) state. The same applies for the $PAcc_P$ (i.e., when $PAcc_N < L_B$) where sending only legitimate CRPs will grow N in Equation (5) and decrease $PAcc_P$, and consequently increase $PAcc_N$, until the qualification of the Medium state is met. Thus, for $\alpha = 0$ CoLAC guarantees returning to the Medium state.

Similar logic is applied when α is dynamically set. By progressively reducing α while being in the High state, the probability that RND exceeds $\alpha.\Delta$, grows. By continuous reduction, α eventually reaches zero and the state diagram becomes what is shown in Figure 4(b). The same effect happens in the Low state when α is reduced. Hence, CoLAC guarantees returning to the Medium state

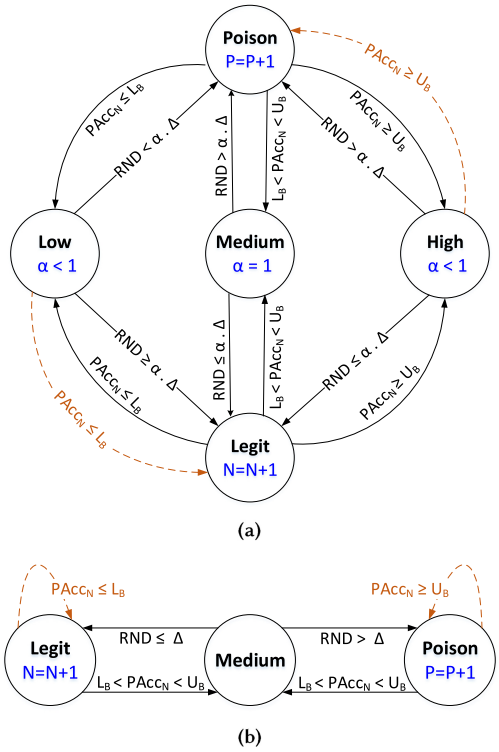


Fig. 4. (a) State transition diagram description of the operation of CoLAC; (b) Reduced state diagram when randomized poisoning is ignored in case of accuracy gets out of bound.

```
1.  Input: αL, αH, U_B, L_B
2.  Δ=0.5; N=1; P=1; PACC_N=0.5;
3.  While (True)
4.      Wait for receiving a challenge;
5.      Generate RND using random number generator;
6.      If (PACC_N > U_B)
7.          State = "High"; α=αH
8.      ElseIf (PACC_N < L_B)
9.          State = "Low" ; α=αL
10.     Else State = "Medium"; α=1
11.     Switch (State)
12.         Case "Low":
13.             If RND ≥ α,Δ;
14.                 State = "Legit"; break;
15.             Else
16.                 State = "Poison"; break;
17.         Case "Medium":
18.             α =1
19.             If (RND ≤ α,Δ)
20.                 State = "Legit"; break;
21.             Else
22.                 State = "Poison"; break;
23.         Case "High":
24.             If (RND > α,Δ)
25.                 State = "Poison"; break;
26.             Else
27.                 State = "Legit"; break;
28.     EndSwitch
29.     If (State == "Poison")
30.         P=P+1; PACC_N = N/(P+N); Send negated response
31.     Else
32.         N=N+1; PACC_N = N/(P+N); Send response
33. EndWhile
```

Fig. 5. Pseudo code summary of the CoLAC approach.

when the value of α is adaptively reduced. Figure 5 shows the pseudo code summary of CoLAC that corresponds to the state diagram in Figure 4(a). □

Lemma #2. *CoLAC achieves its goal regardless of the uniformity of the random number generator.*

Proof. Let's first assume that the distribution of the generated random numbers is known. In such a case, selecting Δ = median would suffice for keeping a balance between poisoned and legitimate CRPs; hence the value of $PAcc_N$ and $PAcc_P$ will be sustained around 50% while being in the Medium state. If the median is unused or unknown, a skewed distribution of random numbers can either have more instances less than Δ or more than Δ. The former would lead to reduced poisoning rate (increasing $PAcc_N$) and could cause transition to the High state. As shown by Lemma 1, an appropriate setting of α will regulate the poisoning rate and enable retransition to the Medium state. Similarly, if the random number generator is biased towards larger numbers than Δ, the frequency of poisoned CRPs dominates and $PAcc_P$ grows. Again, when the increased poisoning rate causes transition to the Low state. Lemma 1 confirms the ability of CoLAC to bring $PAcc_P$ down by increasing the poisoning rate through adjusting α. Thus, CoLAC is robust against variations in the random number distribution. □

Lemma #3. *CoLAC is a lightweight approach that imposes little computational and no communication overhead.*

Table 1. The Randomness Results for the Implemented PUF based on the NIST Test

Test	Passed/ Total	P-Value	Test	Passed/ Total	P-Value	Test	Passed/ Total	P-Value
Frequency	99/100	0.54	FFT	99/100	0.50	Serial	98/100	0.49
Frequency Block	100/100	0.51	Non-Overlap. Template	99/100	0.51	Approx. Entropy	100/100	0.51
Runs	98/100	0.50	Overlap. Template	100/100	0.59	Cumulative Sums	99/100	0.54
The longest Run	99/100	0.49	Universal	5/5	0.99	Random exc.	2/2	0.57
Binary Matrix Ranks	33/33	0.50	Linear Complexity Test	4/4	0.70	Random exc. Var.	2/2	0.40

Proof. CoLAC is designed to suit resource-constrained IoT devices. The poisoning process is simply to toggle the binary response of a PUF, which is a simple negation operation. In an authentication round, the decision for poisoning a response involves: (1) the tracking of the perceived adversary's modeling accuracy, $PAcc_N$, which simply reflects the poisoning rate and constitutes very simple math, and (2) the generation of a random number, which depends on the complexity of the employed PRNG. We note that CoLAC mitigates the effect of PRNG non-uniformity, and thus a lightweight PRNG, e.g., with linear complexity, could be used. Moreover, the PRNG could be implemented in hardware as well. On the other hand, CoLAC uses **both** legitimate and poisoned responses in authenticating the device and hence it imposes no communication overhead. □

6 VALIDATION EXPERIMENTS

To validate the effectiveness of CoLAC, we have implemented a 64-bit Arbiter-PUF on Xilinx ARTIX7 FPGA.

6.1 Test Environment

In our experiments, we have dedicated one FPGA with an embedded PUF to represent the IoT device. The implemented 64-bit Arbiter PUF occupied 243 LUTs and 88 Slice as reported by Xilinx Vivado software. The FPGA board uses the UART protocol for connecting to a PC; the latter plays the role of the server (verifier). The PC generates 40,000 random bit-streams to be used as PUF challenges for the IoT device. The related response is sent back to the PC via the UART. To evaluate the reliability of the proposed architecture in different temperatures, we applied 20,000 randomly generated challenges to the PUF and measured the hamming distance of the responses when a similar challenge is applied. We considered the base temperature as 30°C and repeated the experiments in 0°C, 60°C and 90°C, where on average the discrepancy of responses was 0.65%, 0.92%, and 1.78% in these temperatures, respectively. This demonstrates the reliability of our design. Moreover, the noise effect in the same temperature resulted in a negligible (0.25%) discrepancy in response, which confirms the robustness of our implementation.

The randomness of the implemented 64-bit Arbiter-PUF was evaluated using 15 statistical tests, published by NIST for assessing the randomness of true random generators [6], with 5,000,000 randomly selected challenge bit-streams. The responses were divided into 100 blocks each including 50,000 responses, and we applied the NIST tests to each block. Note that some of the tests (e.g., Universal) need larger blocks, hence we partitioned our responses accordingly. As shown in Table 1, our PUF structure has passed almost all tests, which confirms the randomness of our implemented PUF. To show that our proposed scheme does not change the randomness of the transferred responses, we have applied the NIST tests to the transferred responses after applying CoLAC, i.e., the poisoned responses generated by CoLAC. The results are shown in Table 2 and are very similar to those in Table 1; confirming the randomness of the CoLAC- generated responses.

Table 2. The Randomness Results for the Poisoned PUF Responses based on the NIST Test

Test	Passed/ Total	P-Value	Test	Passed/ Total	P-Value	Test	Passed/ Total	P-Value
Frequency	99/100	0.54	FFT	99/100	0.50	Serial	98/100	0.49
Frequency Block	100/100	0.51	Non-Overlap. Template	98/100	0.49	Approx. Entropy	100/100	0.51
Runs	98/100	0.50	Overlap. Template	100/100	0.59	Cumulative Sums	99/100	0.54
The longest Run	99/100	0.49	Universal	5/5	0.99	Random exc.	2/2	0.57
Binary Matrix Ranks	33/33	0.50	Linear Complexity Test	4/4	0.70	Random exc. Var.	2/2	0.40

To assess the power overhead, the PUF was isolated from the underlying circuit; the power consumption of a 64-bit PUF with 16 response bits was the measured by the Xilinx Power Estimator tool and found to be 0.002W. We have considered an adversary that intercepts all exchanged CRPs and pursues the following two attack scenarios: (i) applying the PUF modeling scheme of [32], and (ii) employing a machine learning scheme to recognize the poisoning pattern given partial knowledge of correct and erroneous CRPs. The latter opts to assess CoLAC's ability of obscuring the poisoning pattern. The validation experiments and results are discussed in the rest of the section. We employed the SVM, **Logistic Regression (LR),** and NN as representatives of ML techniques that an adversary pursues to conduct a modeling attack against the deployed PUFs. Note that the CMA-ES scheme that has been proposed in [7] to model the PUF based on the sensitivity of its response to environmental noise, e.g. temperature or voltage variations, is not applicable in our case since: (i) an eavesdropper doesn't have physical access to the PUF to repeat the same query multiple times to benefit from the measurement noise in PUF modeling, instead intercepting the exchanged CRPs is the most viable means, and (ii) even if the PUF is queried with the same challenge once a while, the node's response to the same query may change due to poisoning. This confuses the adversary as CMA-ES may consider the poisoned data as measurement noise.

6.2 Experiment Setup

In order to study the performance of CoLAC, we have conducted three experiments. For the three experiments, we have considered both uniform random generator and non-uniform Gaussian distribution. For the latter we considered three variants with the same $\mu = 0.2$, and $\sigma = 0.1$, 0.3, and 0.7. We note that the Gaussian distribution is trimmed at 0 and 1 and that the three configurations have different skewness and kurtosis. The first experiment is for measuring the accuracy of the adversary's PUF modeling, where we divide the CRP dataset captured by the eavesdropper randomly into two subsets for supporting the training and test phases. We take the average of the accuracy for 30 different runs. For the second experiment, we capture the effect of the size of the dataset used by the adversary in modeling the PUF. Such an experiment opts to gauge CoLAC robustness overtime as the adversary potentially intercepts more CRPs.

For the third experiment, we assess the ability of the adversary to detect the poisoning pattern where the CRPs in the training set are labeled as correct and erroneous. Although such a scenario is not practical given the difficulty for the eavesdropper to distinguish between poisonous and legitimate data, we opt to assess the resilience of CoLAC even under such a rare scenario. The idea is to assess the adversary's ability to predict whether the response at time epoch t is legitimate or poisonous, when knowing the classification of the CRPs in the previous n time epochs. To detect the pattern, we use an LSTM that is trained for 100 epochs with batch size of 1024 CRPs. The LSTM is fed with a sequence of numbers of consecutive CRPs labeled as legitimate or poisoned.

Basically, we have labeled the binary values representing responses as correct "C" and erroneous "E", and converted them into a sequence that reflects the lengths of legitimate and poisoned runs. For example, for the sequence "CCCEECEEE" we feed the LSTM with "3, 2, 1, 3" where the number indicates the count of correct or erroneous responses. After training the LSTM, we start predicting the upcoming pattern until we have our entire test dataset. Then, the prediction accuracy is used to indicate the attacker's ability to infer the poisoning pattern. We again note that the use of LSTM is not for modeling the PUF, but rather for predicting the poisoning pattern; the LSTM is applied only as a means for validating CoLAC. We also compare the performance of our lightweight approach with a version where an elaborate label flipping optimization is applied to determine what challenge to use next, as will be explained in the following.

6.3 Challenge Selection Optimization

CoLAC pursues a label flipping AML strategy. As pointed out in Section 4, a label flipping strategy could be applied with and without optimizing the challenge selection. Ideally, each time data poisoning is to take place, an optimal CRP is picked to inflict the most degradation on the adversary's model. However, this optimization would entail excessive reformation of the ML model by the server based on how frequently it is done and the solution domain (number of all possible CRPs); clearly such an approach would impose undesirable overhead. Therefore, CoLAC pursues random selection of challenge bit streams in order to limit the overhead. To gauge the effect of such a decision on performance, we have studied the impact of the challenge selection optimization when incorporated within CoLAC. In the experiment, we consider a variant of CoLAC in which CRPs within a certain time window are selected for each poisoning and legitimate transmission. Basically, considering the set of challenges $\Theta = \{C_1, \ldots, C_m\}$ that are to be used in the next m iterations, i.e., authentication rounds, we determine an exact order of the challenges to be transmitted. The responses for these challenges may be poisoned when both the server and the IoT device decide to do so, according to the state diagram in Figure 4(a). To elaborate, we define the ordered list $\Psi = \{\psi_1, \ldots, \psi_m\}$ of the challenges *in* Θ. In essence, the list Ψ represents the sequence of challenges sent by the server in the next m iterations, i.e., ψ_i, reflects the challenge used in the i^{th} authentication round. The ordering criterion is the current PUF modeling accuracy and the impact of the challenge on the loss function. For the former, we note that $PAcc_N$ and $PAcc_P$ are players; therefore we strive to minimize the difference between them which implies becoming closer to 50% overall modeling accuracy. Let σ_i be $|PAcc_N - PAcc_P|$ at the i^{th} round. Based on Equations (3) and (4), $\xi = |\frac{N-P}{N+P}|$. Given that ξ is used for ranking the challenges in Θ, we simply take $\xi = |N - P|$. Let δ_i be the accuracy loss, introduced by challenge $C_i \in \Theta$. The label flipping optimization is then to assign the challenge with the highest loss to the authentication round with the largest σ; this is concisely captured by:

$$(\downarrow^i | \nexists C_\perp j \in (\wedge \equiv \left[\left[\sigma_\downarrow i < \sigma_\downarrow j \Longmapsto \delta_{\downarrow_j} < \delta_{\downarrow_i}, \forall 1 \leq i \leq m\right]\right], \quad 1 \leq j \leq m \tag{6}$$

$$\psi_i | \nexists C_j \in \Theta \wedge \sigma_i < \sigma_j \Longmapsto \delta_j < \delta_i, \ \forall 1 \leq i \leq m, \ 1 \leq j \leq m \tag{7}$$

Note that the challenge selection optimization is done only at the server side and the IoT device is not engaged at all. The steps go as follows. First, the set Θ is formed; here the selection of $C_i \in \Theta$ can be random within the unused combinations of challenge bit patterns. For each $C_i \in \Theta$, δ_i is calculated using the loss function when a neural network is applied to model the PUF. The entries $C_i's$ of set Θ are then sorted in a descending order according to δ_i. Second, CoLAC follows the state diagram in Figure 4(a) for m consecutive times; for each the corresponding σ is calculated. We note that no communication with the IoT device takes place and σ is estimated offline. To form Ψ, we iterate sequentially on all entries of the sorted Θ, where the top entry is assigned to the

iteration with the largest σ, and the second entry in Θ is assigned to the iteration with the largest σ among the rest, and so on. This process is repeated for each of the m challenges in Θ, and in effect defines all ψ_i. The set Ψ specifies what challenges the server will use in the next m iterations. In the experiment, m is set to 40,000.

6.4 Performance Metrics and Parameters

The following metrics are used to assess the performance:

- *Accuracy of predicting the PUF response*: It reflects the adversary's ability in guessing the correct response through PUF modeling either at an early stage or over time. To calculate the accuracy, we use SVM, NN, and LR, as stated earlier. Nonetheless, we have observed that there is not much difference among the accuracies achieved by these three LM techniques. Therefore, most of the reported results in this section are based on NN in order to avoid redundancy. The used neural network model has three hidden nonlinear layers and one output linear layer. For three nonlinear layers, we employ the **Rectified linear unit (ReLU)** as the activation function with 5, 10, and 15 neurons, respectively. The learning rate and momentum are 0.01 and 0.99, respectively. The number of epochs is set to 3,000. With this network network architecture, a 64 bits Arbiter PUF is modeled with the accuracy of 98% using 2,000 CRPs for training.
- *Accuracy of predicting the poisoning pattern*: This assesses the probability of defeating CoLAC by guessing the poisoning pattern and consequently classifying correct and erroneous CRPs. An LSTM is used for assessing the poisoning pattern prediction accuracy, as explained earlier.

We also capture the effect of the following key parameters:

- Δ: This reflects the probability of injecting poisonous data using the random sequence.
- $Bound(\beta)$: This refers to the upper and lower accuracy bound from the ideal accuracy. Where $L_B = min(1-\beta, \beta)$ and $U_B = max\ (\beta, 1-\beta)$, which implies a symmetric bound for $PAcc_N$ and $PAcc_P$.
- *The size of the intercepted CRP set*: Given that the adversary applies data-driven attack strategies, the size of training and test sets impacts the attack success.
- α: The setting of such a regulator affects the accuracy stability and convergence to safe state. Here we try static and dynamic settings. In the former we experiment with $\alpha = 0$, and 0.3. In the dynamic scenario, we adaptively reduce α by a factor of $30(PAcc_N - U_B)$ when staying in the High state or $30(L_B - PAcc_P)$ when staying Low state. Such a factor is applied after each CRP exchange in the respective state.

We are reporting the PUF modeling accuracy for the following scenarios that are also summarized in Table 3:

- *CoLAC-U*: This reflects the case when the PRNG produces uniformly distributed random numbers between 0 and 1. In such a case setting Δ to 0.5 (median) will target achieving a 50% poisoning rate. We study a variant, which we refer to as "CoLAC-*U*-β-α". Such a variant simply becomes a mechanism that poisons when violating the desired bounds and the generated random number exceeds Δ, regulated by α to control the poisoning rate.
- *CoLAC-N*: This is used as a prefix to indicate the non-uniformity of the random number distribution. Under this setting, both the accuracy bounds, Δ, and α play significant roles. Therefore, we show the results for fixing Δ at 0.5 to capture the effect of non-uniformity (denoted as "*CoLAC-N*"); in this case no accuracy bounds are observed. We also study the

Table 3. Summary of the Compared Configurations of CoLAC

	Uniform PRNG	β	Δ	α	AML
CoLAC-U	Yes	No	Yes	No	No
CoLAC-U-β-α	Yes	Yes	Yes	Yes	No
CoLAC-N	No	No	Yes	No	No
CoLAC-N-β-α	No	Yes	Yes	Yes	No
CoLAC-N-β-α-AML	No	Yes	Yes	Yes	Yes
Bound-only	N/A	Yes	No	No	No

performance while varying the accuracy bounds with and without AML-based challenge selection. We use "β-α", and "β-α-*AML*", as postfixes to denote these two cases, respectively. The objective is to show that the accuracy bound is a means for mitigating the effect of non-uniform random number distributions.

- *Bound-only*: For this case, the poisoning pattern is not randomized. When the accuracy of the attacker's model exceeds the bound, poisoning takes place. The bound is set to 0.5, which is the ideal accuracy for a binary classifier, implying that the adversary will have no fidelity in predicting the PUF response.

6.5 Experimental Results

The first experiment opts to assess the effectiveness of CoLAC as a countermeasure by tracking the accuracy of the attacker's PUF model. Such an accuracy is measured by applying the machine learning attack of [32].

Effect of Δ: Figure 6 shows the attack accuracy for different values of Δ, which influences the poisoning rate. Here we are assuming $\alpha = 1$, and just using Δ to control the poisoning rate. For *CoLAC-U*, the value of Δ constitutes the poisoning rate, and the accuracy simply equals max [Δ, and (1-Δ)]. Therefore, the accuracy takes the shape of a parabola with vertex at $\Delta = 0.5$, and the corresponding accuracy also is 0.5. Thus, for a uniform PRNG, matching the vertex reflects the ideal scenario in terms of safeguarding the PUF against modeling attacks. The results in Figure 6 also capture the effect of non-uniformity on the performance where a Gaussian-based PRNG is used; three configururations are tried to gauge the effect of skewness and kurtosis. The variation of the accuracy profiles among the three non-uniform distributions clearly point out the importance of regulating the poisoning process and not relying on the uniformity of the PRNG. In particular, the case of $\sigma = 0.1$ demonstrates the potential rise of the model accuracy if one relies only on Δ. To elaborate, setting $\Delta = 0.5$, which is optimal for uniform PRNG, could instead lead to model accuracy of 98% when $\sigma = 0.1$. Similar conclusion can be made when SVM and LR are used. Figure 6 confirms the importance of not relying on the PRNG uniformity and the incorporation of α as a mitigation measure. We factor in α and use $\Delta = 0.5$ for the other experiments discussed in the balance of this section, unless mentioned otherwise.

Effect of α: Figure 7 highlights the role of α as a regulator while setting $\Delta = 0.5$. The results are based on NN, and split among two plots to improve readability. Figure 7(a) compares three settings for α when a non-uniform PRNG is employed. The results are for a Gaussian PRNG with $\sigma = 0.3$ under varied accuracy bounds (β). As noted earlier in the section, the desired accuracy bound is max [β, (1-β)]; thus for a bound of 0.2, the accuracy should not exceed 0.8. As indicated by the plots, the accuracy indeed stays below the bound, with some exception for large α. Compared to the plot for *CoLAC-N* ($\sigma = 0.3$) in Figure 6, incorporating α regulates the shape of the curve and makes it close to the case of uniform PRNG, i.e., a parable shape; this applies for all three

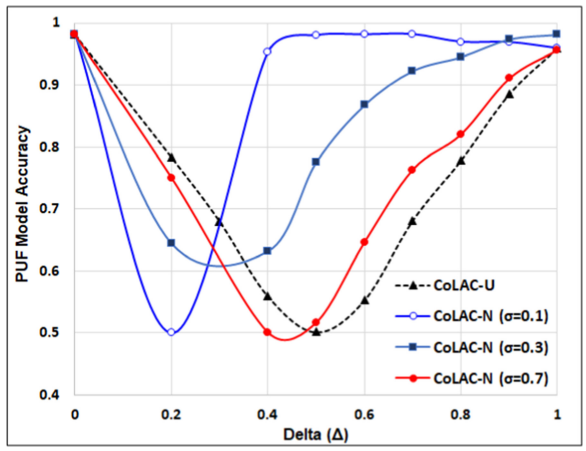
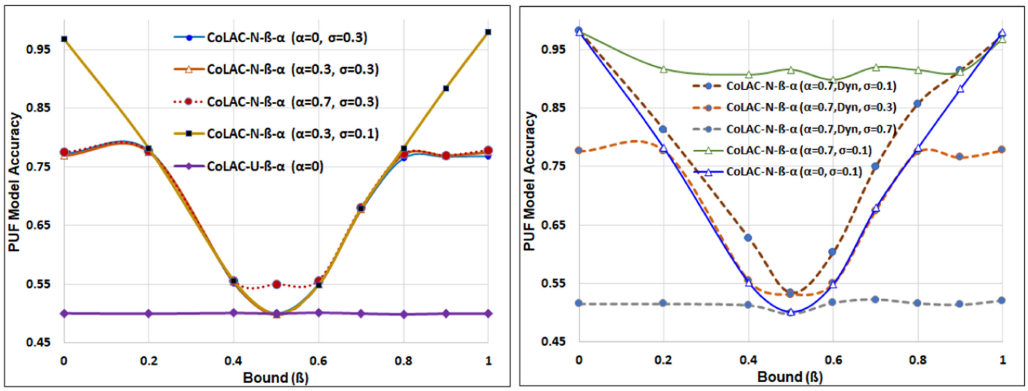


Fig. 6. Capturing the effect of Δ on the poisoning rate, and consequently, on the accuracy when using NN to the model PUF. Here, poisoning takes place when RND exceeds Δ. We are showing the results for both uniform and non-uniform PRNGs. The latter in essence follows a Gaussian distribution with $\mu = 1$; we are considering three different configurations of such a non-uniform PRNG.



(a) (b)

Fig. 7. Evaluating the role of α in regulating the poisoning process relative to the accuracy bounds, while (a) α is fixed, and (b) α dynamically adjusted over time to accelerate bringing down the accuracy to within the desired bounds. The effect of PRNG skewness and kurtosis is also captured.

considered values of α. Having a relatively large α, i.e., 0.7, affects the achievable accuracy; this is clear when focusing on β in the range [0.4, 0.6], where the accuracy does not go down to 0.5 when $\beta = 0.5$. Such an observation is consistent with the motive of introducing small α, to deal with non-uniformity and allow the accuracy to be controlled in the Low and High states. Later in this section we will show the effect of α on detectability of the poisoning pattern, and point out that too small values of α are not recommended. Hence, setting $\alpha = 0.3$ will be preferred over $\alpha = 0$, although both yield the same results in Figure 7(a).

Figure 7(a) also shows the results when incorporating α while employing a uniform PRNG. We are showing only the case for $\alpha = 0$ since the curves for $\alpha = 0.3$ and 0.7 are similar. Basically, having $\Delta = 0.5$ suffices for keeping the accuracy within bounds and α does not play much a role. This is consistent with the results in Figure 6. Figures 6(a) and 6(b) study the effect of kurtosis

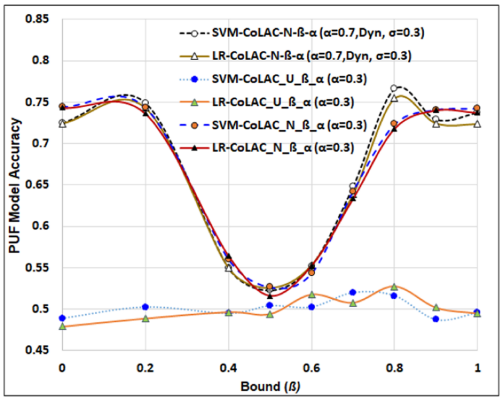


Fig. 8. The SVM and LR results for evaluating the role of α in regulating the poisoning process relative ß.

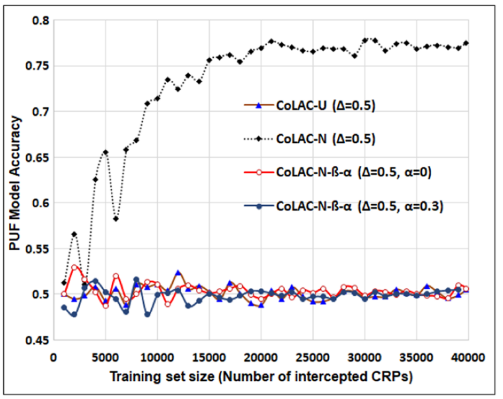
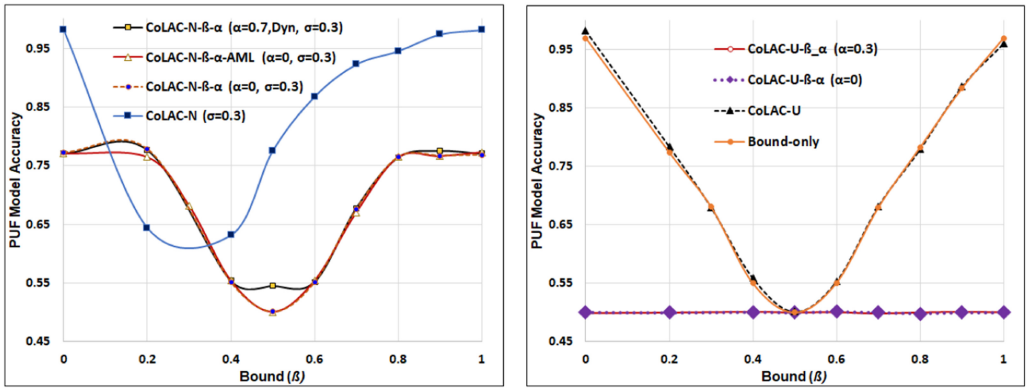


Fig. 9. Capturing the effect of the size of the training dataset on the effectiveness of CoLAC for Δ = 0.5.

of the non-uniform PRNG. Figure 7(a) compares the results when σ changes from 0.3 to 0.1; the latter boosts the kurtosis of the distribution. The results when $\alpha = 0.3$, indicate that the kurtosis makes it more challenging to keep the accuracy within bound; yet CoLAC is succeeding in doing so for this configuration. That is not the case when $\alpha = 0.7$, as indicated by the *CoLAC-N-ß-α (α = 0.7, σ = 0.1)* curve in Figure 7(b). Basically, with high kurtosis (small σ), the accuracy sometimes exceeds the bound and thus small α should be used.

In addition, Figure 7(b) opts to capture the impact of dynamic setting of α. Since α is influential only under non-uniformity conditions, we consider three Gaussian PRNGs with σ = 0.1, 0.3, and 0.7. In this experiment, α starts at 0.7 and is reduced gradually by a factor proportional to the deviation from the bound, every time the condition relating RND and Δ is not met. As demonstrated by the results, adapting the setting of α is quite effective in dealing with skewness and low kurtosis, where the results for higher σ almost matches the ideal case for a uniform PRNG. With high kurtosis (small σ), the accuracy sometimes slightly exceeds the bound. However, compared with the case of fixed α for the same Gaussian PRNG, i.e., *CoLAC-N-ß-α* ($\alpha = 0.7$, $\sigma = 0.1$), dynamic adjustment of α is indeed effective in observing the desired accuracy bound and is deemed a must for PRNG with very high kurtosis. Considering the results of Figure 7 collectively, one can conclude that picking a small α is crucial for controlling the accuracy while using non-uniform PRNG. Dynamic adjustment of α helps in dealing skewness and high kurtosis. In Figure 8, we show the results when applying SVM and LR. Comparing the curves in Figure 8 to the corresponding ones for NN in Figure 7 demonstrates the very high similarity of the accuracy achieved by the three ML techniques. Therefore, we show only the results of NN in the balance of this section.

Effect of Training Dataset: Figure 9 depicts how the attacker's modeling accuracy is affected by the size of the training dataset. We show the results for *ColAC-U* and *CoLAC-N* with $\Delta = 0.5$ and $ß = 0.5$. The results for non-uniform PRNG are based on Gaussian distribution with $\mu = 0.2$ and $\sigma = 0.3$. Overall, CoLAC consistently achieves excellent results by keeping the adversary's accuracy to around 0.5 when the generated random numbers are uniformly distributed and by employing the regulator α to cope with non-uniformity. CoLAC sustains its superior performance even when the attacker uses more data to train the PUF model. The results show that relying only on Δ is not sufficient defense as the non-uniformity of PRNG could allow accuracy to get out of hand; with increased training the PUF model accuracy improves, as indicated by the *CoLAC-N*



(a) (b)

Fig. 10. Capturing the effect of bounds and the various CoLAC parameters on the PUF modeling accuracy with (a) non-uniform and (b) uniform PRNGs.

curve. The regulator α helps in controlling the accuracy as demonstrated by the *CoLAC-N-ß-α* curves.

Collective Effect of Parameters: Figure 10 compares the various configurations for CoLAC, while fixing Δ to 0.5. Figure 10(a) considers non-uniform Gaussian PRNG with $\mu = 0.2$ and $\alpha = 0.3$. For *CoLAC-N*, no bounds are specified, while the other configurations factor in $ß$ in the process. The plot for *CoLAC-N* is similar to that in Figure 6 and is included here for ease of comparison. When employing $\alpha = 0$, *CoLAC-N-ß-α* observes the desired accuracy bound. In such configuration, the challenge bit stream is randomly selected. When dynamically varying α, the performance stays mostly good, except around $ß = 0.5$, i.e., when the bounds are very tight on the accuracy. While setting $\alpha = 0$ in the Low and High states appears to be more effective in such a case, it could expose the defense strategy to potential poisoning pattern detection as later discussed.

The *CoLAC-N-ß-α*-AML curve in Figure 10(a) reports the results when picking the challenge based on AML optimization; here α is set to 0 as well. As in the figure, the results almost match those of without the challenge selection optimization, i.e., *CoLAC-N-ß-α* ($\alpha = 0$); only slight improvement is observed when $ß \geq 0.8$ or $ß \leq 0.2$. We conclude that *CoLAC-N-ß-α* is a better choice since it imposes very negligible overhead. Thus, leaving out challenge selection through AML optimization is indeed justified.

Figure 10(b) reports the results when a uniform PRNG is used. As noted earlier, α is not a player in this case where the performance of *CoLAC-U-ß-α* for $\alpha = 0$ and 0.3 stays the same, where the accuracy depends mostly on Δ. The figure shows the results for *Bound-only*, as well. For *CoLAC-U* the value of Δ is equal to $ß$ in this plot, and hence the performance is similar to *Bound-only* since they both are dependent on the poisoning rate. We note that the *Bound-only* configuration involves deterministic poisoning based on the bound, and hence it is susceptible to poisoning pattern detection attack as we show when discussing the third experiment. In summary, the results confirm that CoLAC is capable of controlling the accuracy of the adversary classifiers, regardless of the uniformity of the random number generator.

Poisoning Pattern Detectability: Figure 11 illustrates the adversary's inability to detect the poisoning pattern and consequently negating the effect of CoLAC. Again in this experiment, the adversary is assumed to have known some legitimate and poisonous CRPs and builds on such knowledge to predict the poisoning pattern. In the figure, we show the results for various

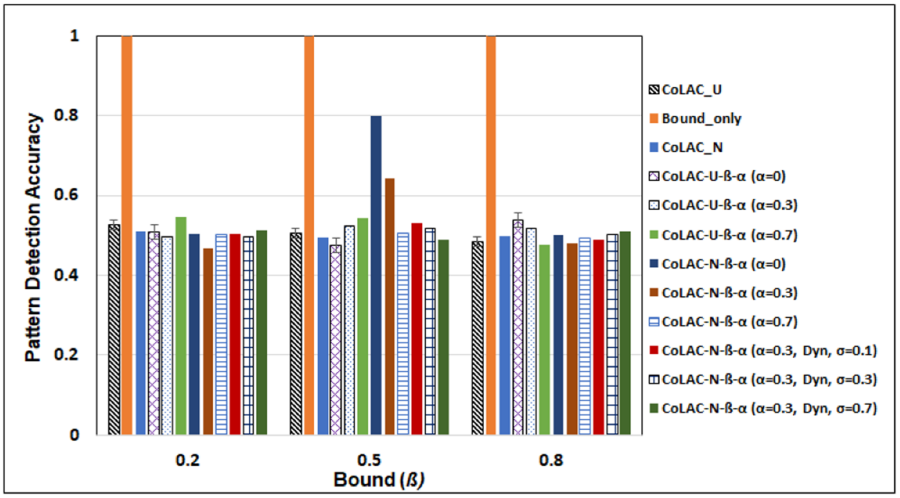


Fig. 11. Capturing the effect of α and the accuracy bound (ß) on the detectability of the poisoning pattern by the attacker.

Table 4. Comparing the Baseline PUF with the Protected Counterpart (post COLAC Application) using PUFmeter Tool [43]

Metrics and Techniques	Unprotected PUF (Baseline)	Protected PUF (CoLAC)
Average Sensitivity	0.489	0.3614
Noise Sensitivity	0.211	0.183
K(Parameter in K-junta testing)	0	0
LMN algorithm	N/A	N/A

configurations and compare the cases when poisoning is controlled by only Δ (*CoLAC-U*), only the bound (*Bound-only*), and both of them (*CoLAC-U-ß-α*), while employing a uniform random number generator. We also report the case when a non-uniform random number generator is used (*CoLAC-N*). The results demonstrate that relying on the poisoning rate alone, i.e., the case of *Bound-only*, can make the poisoning pattern 100% predictable. The results also confirm the importance of the randomization pursued by CoLAC for obscuring the pattern, where detectability stays around 50%.

As pointed out earlier, α is not a player when the PRNG is uniform; yet it is very influential for the variants of *CoLAC-N*. Indeed for α equal zero the poisoning pattern is deterministic and the detectability is significantly high for *CoLAC-N-ß-α*; a larger α will be better in that context (see the bars for $\alpha = 0.3$, and $\alpha = 0.7$). Recall that a small α is better for sustaining low accuracy; therefore, there is a tradeoff. However, when the accuracy bounds are flexible, i.e., not narrowed around 0.5, the detectability of the pattern is not high except for *Bound-only* due to lack of randomization; such a performance is expected since α does not play a role except when exceeding the bound. Again, this experiment assumes that the adversary could intercept and distinguish legitimate from poisoned CRPs, which is in itself unrealistic. Nonetheless, CoLAC performs quite robustly and prevents the adversary from predicting the poisoning pattern over time.

Resiliency against K-junta and Fourier based attacks: We used the PUFmeter tool to assess the resiliency of the PUF against K-junta and Fourier based attacks discussed in [43]. Table 4 shows the results, reporting four metrics that gauge the robustness of a PUF, where the results for the

64-bit baseline Arbiter-PUF we used in the validation are shown with and without applying CoLAC. These results have been extracted for the accuracy level (ε) = 0.05 and confidence level (delta) = 0.01, and desired reliability = 0.01. The PUFmeter tool output indicates that with CoLAC the PUF is resilient to K-junta testing, and Fourier based attacks.

7 CONCLUSIONS

This paper has presented an adversarial machine learning based methodology to safeguard Physically Unclonable Functions (PUFs) against modeling attacks. By intercepting exchanged challenge-response pairs (CRPs), an eavesdropper can employ machine learning to model the PUF and predict its output. This constitutes a serious threat as PUFs are used for providing critical security services for IoT frameworks such as authentication. We have presented a novel and effective countermeasure, namely, CoLAC, which applies an adaptive data poisoning strategy that factors in potentially leaked information, and injects randomness in the poisoning pattern. The approach is lightweight and allows implicit coordination and prediction of poisoned CRP exchanges. Specifically, the same pseudo random number generator is employed at the two communicating parties in order to ensure synchronization and consensus between them.

CoLAC has been validated using datasets generated by a PUF implementation on an FPGA. The validation results have demonstrated CoLAC's effectiveness in degrading the attacker's modeling accuracy, and its robustness against variations in the random number distributions. In summary, if the designer can guarantee the uniformity of the employed PRNG setting Δ to 0.5 would suffice for countering PUF modeling attacks. However, such a setting is heavily reliant on the PRNG; instead it is recommended to factor in both Δ and α. Setting α to between 0.3 and 0.5 would strike a balance between robustness against modeling attacks and potential poisoning pattern detection. The results also pointed out the advantage of adaptively adjusting the value of α to cope with skewness and kurtosis of random number distribution. Finally, random selection of challenge bit streams would yield high protection and an elaborate label flipping optimization is unwarranted. In the validation, we have confirmed the resilience of CoLAC against three prominent ML techniques; As a future extension we plan to explore theoretical frameworks, e.g., [17], to analytically verify the resilience of our scheme.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. 2015. Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials* 17, 4 (2015), 2347–2376, 4th Quart.

[2] M. Aman, M. H. Basheer, and B. Sikdar. 2019. Data provenance for IoT with light weight authentication and privacy preservation. *IEEE Internet of Things Journal* 6, 6 (2019), 10441–10457.

[3] M. Aman, K. C. Chua, and B. Sikdar. 2017. Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet of Things Journal* 4, 5 (2017), 1327–1340.

[4] M. Aman, K. C. Chua, and B. Sikdar. 2016. Physical unclonable functions for IoT security. *Proc. Int'l Workshop on IoT Privacy, Trust, and Security* (2016), 10–13.

[5] M. Barbareschi et al. 2018. A PUF-based hardware mutual authentication protocol. *Journal of Parallel and Distributed Computing* 119, (2018), 107–120.

[6] L. E. Bassham et al. 2010. *A statistical test suite for random & pseudorandom number generators for cryptographic applications. NIST SP 800-22*, National Institute of Standards and Technology.

[7] G. T. Becker. 2015. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin.

[8] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay. 2017. A PUF-based secure communication protocol for IoT. *ACM Trans. Embed. Comput. Syst* 16, 3 (2017), 67:1–67:25.

[9] U. Chatterjee et al. 2019. Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database. *IEEE Transactions on Dependable and Secure Computing* 16, 3 (2019), 424–437.

[10] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. 2014. Secure lightweight entity authentication with strong PUFs: Mission impossible? *Proc. Cryptographic Hardware and Embedded Systems (CHES)*. 451–475.

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

[12] E. Dubrova et al. 2019. CRC-PUF: A machine learning attack resistant lightweight PUF construction. *Proc. of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 264–271.

[13] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni. 2019. A survey of Internet of Things (IoT) authentication schemes. *Sensors* 19 (2019), 1141–1183.

[14] F. Farha et al. 2020. SRAM-PUF based entities authentication scheme for resource-constrained IoT devices. *IEEE Internet of Things Journal* 1–1.

[15] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. 2008. Controlled physical random functions and applications. *ACM Trans. Inf. Syst. Secur.* 10, 4 Article 3, (2008), 22.

[16] I. J. Goodfellow, S. Jonathon, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*. San Diego, CA, USA, May 2015.

[17] F. Ganji et al. 2020. Rock'n'roll PUFs: Crafting provably secure PUFs from less secure ones (extended version). *J Cryptogr Eng*.

[18] P. Gope et al. 2018. Lightweight and practical anonymous authentication protocol for RFID systems using physically unclonable functions. *IEEE Transactions on Information Forensics and Security* 13, 11 (2018), 2831–2843.

[19] P. Gope and B. Sikdar. 2018. Lightweight and privacy-preserving two-factor authentication scheme for IoT devices. *IEEE Internet of Things Journal* 6, 1 (2019), 580–589.

[20] C. Gu, C.-H. Chang, W. Liu, S. Yu, Q. Ma, and M. O'Neill. 2019. A modeling attack resistant deception technique for securing PUF based authentication. *Proc. Of the Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Xi'an, China.

[21] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas. 2014. Physical unclonable functions and applications: A tutorial. *Proceedings of IEEE* 102, 8 (2014), 1126–1141.

[22] C. Huth, J. Zibuschka, P. Duplys, and T. Güneysu. 2015. Securing systems on the Internet of Things via physical properties of devices and communications. *Proc. of the Annual IEEE Systems Conference (SysCon)*. Vancouver, BC (2015), 8–13.

[23] N. Karimi, J.-L. Danger, and S. Guilley. 2018. Impact of aging on the reliability of delay PUFs. *Journal of Electronic Testing* 34, 5 (2018), 571–586.

[24] Y. Lao, B. Yuan, C. H. Kim, and K. K. Parhi. 2017. Reliable PUF-based local authentication with self-correction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 36, 2 (2017), 201–213.

[25] Q. Ma, C. Gu, N. Hanley, C. Wang, W. Liu, and M. O'Neill. 2018. A machine learning attack resistant multi-PUF design on FPGA. *Proc. of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jeju, Korea (South), 2018, 97–104.

[26] R. Maes, A. Van Herrewege, and I. Verbauwhede. 2012. Puiky: A fully functional PUF-based cryptographic key generator. *Cryptographic Hardware and Embedded Systems (CHES'12)*, 302–319.

[27] M. H. Mahalat, S. Saha, A. Mondal, and B. Sen. 2018. A PUF based light weight protocol for secure WiFi authentication of IoT devices. In *the Proceedings of the 8th International Symposium on Embedded Computing and System Design (ISED)*. Cochin, India 2018, 183–187

[28] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas. 2012. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. *Proc. of the IEEE Symposium on Security and Privacy Workshops*. San Francisco, CA, (2012), 33–44.

[29] P. H. Nguyen et al. 2019. The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), 4, 243–290.

[30] M. A. Qureshi and A. Munir. 2020. PUF-IPA: A PUF-based identity preserving protocol for Internet of Things authentication. In *IEEE Annual Consumer Communications Networking Conference (CCNC)* 2020, 1–7.

[31] M. A. Qureshi and A. Munir. 2021. PUF-RAKE: A PUF-based robust and lightweight authentication and key establishment protocol. *IEEE Transactions on Dependable and Secure Computing*, 2021 (to appear) DOI: https://doi.org/10.1109/TDSC.2021.3059454

[32] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. *Proc. of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, 237–249.

[33] K. Sha, W. Wei, T. A. Yang, Z. Wang, and W. Shi. 2018. On security challenges and open issues in Internet of Things. *Future Generation Computer Systems* 83 (2018), 326–337.

[34] G. E. Suh and S. Devadas. 2007. Physical unclonable functions for device authentication and secret key generation. *Proc. of the 44th ACM/IEEE Design Automation Conference (DAC)* 9–14, 2007.

[35] E. I. Vatajelu, G. Di Natale, M. S. Mispan, and B. Halak. 2019. On the encryption of the challenge in physically unclonable functions. *Proc. of the IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS).* Rhodes, Greece.

[36] J. R. Wallrabenstein. 2016. Practical and secure IoT device authentication using physical unclonable functions. *Proc. of the IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)* (2016), 99–106.

[37] S.-J. Wang, Y.-S. Chen, and K. Li. 2019. Adversarial attack against modeling attack on PUF. *Proc. of the 56th ACM/IEEE Design Automation Conference (DAC).* Las Vegas, NV, USA.

[38] N. Wisiol et al. 2020. Splitting the interpose PUF: A novel modeling attack strategy. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), 97–120.

[39] H. Xiao, H. Xiao, and C. Eckert. 2012. Adversarial label flips attack on support vector machines. *Proc. the 20th European Conference on Artificial Intelligence (ECAI),* Montepellier, France August 2012.

[40] M.-D. Yu et al. 2016. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems* 2, 3 (2016), 146–159.

[41] M. D. M. Yu, D. M'Rai'hi, I. Verbauwhede, and S. Devadas. 2014. A noise bifurcation architecture for linear additive physical functions. *Proc. of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* 124–129.

[42] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang. 2019. Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response. *IEEE Transactions on Information Forensics and Security* 14, 4 (2019), 1109–1123.

[43] F. Ganji et al. 2019. PUFmeter a property testing tool for assessing the robustness of physically unclonable functions to machine learning attacks. *IEEE Access* 7 (2019), 122513–122521.

[44] S.-J. Wang, Y.-S. Chen, and K. S.-M. Li. 2021. Modeling attack resistant PUFs based on adversarial attack against machine learning. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11, 2 (2021), 306–318.

[45] M. Ebrahimabadi, W. Lalouani, M. Younis, and N. Karimi. 2021. Countering PUF modeling attacks through adversarial machine learning. *Proc. ISVLSI,* 2021, 356–361.

Received April 2021; revised September 2021; accepted September 2021