# A PUF-Based Modeling-Attack Resilient Authentication Protocol for IoT Devices

Mohammad Ebrahimabadi, *Graduate Student Member, IEEE*, Mohamed Younis, *Senior Member, IEEE*, and Naghmeh Karimi, *Member, IEEE*

***Abstract*—Physical unclonable functions (PUFs) offer a promising solution for the authentication of Internet of Things (IoT) devices as they provide unique fingerprints for the underlying devices through their challenge–response pairs. However, PUFs have been shown to be vulnerable to modeling attacks. In this article, we propose a novel protocol to thwart such vulnerability by limiting the adversary's ability to intercept the whole challenge bits exchanged with IoT nodes. We split the challenge bits over multiple messages and engage one or multiple helper nodes in the dissemination process. We further study the implications of various parts of the challenge patterns on the modeling attack and propose extensions of our protocol that exploit bits scrambling and padding to ameliorate the attack resiliency. The experimental results extracted from a 16-bit and a 64-bit arbiter-PUF implemented on FPGA demonstrate the effectiveness of the proposed methods in boosting the robustness of IoT authentication.**

***Index Terms*—Authentication, Internet of Things (IoT), physical unclonable function (PUF), machine learning (ML).**

## I. Introduction

THE NOTION of the Internet of Things (IoT) has emerged to characterize the internetworking of numerous and diverse devices to form ubiquitous computing systems that enable probing the environment, sharing data, and controlling physical processes. In essence, an IoT provides a core infrastructure that extends the communication and exchange of data from servers, personal computers and smartphones to an enormous range of objects used in everyday life. IoT applications can be found in several domains, e.g., scientific, military, and civil domains. For example, in space applications, an IoT would enable broad accessibility at the global scale by internetworking of space assets owned and operated by independent entities. Similarly, in the realm of smart cities, the Internet of Vehicles would self-manage traffic on the road through interaction between vehicles and cooperation with road infrastructure, e.g., traffic signals. Overall, it is estimated that 100 billion devices will be interconnected through IoT frameworks by 2025 [1].

The societal impact and role of IoT elevate the importance of guarding it against security threats. However, countering security attacks in IoT is more challenging than in traditional networks due to the wide range of communication protocols and limited capabilities of the involved devices. Security threats for IoT devices range from enforcing malicious malfunctions and denial of service to leaking sensitive information. Given the role that an IoT plays, combating security threats is a must, and provisioning for node authentication, data integrity, access control, and privacy would be expected in the design [2]–[5]. However, in such an era of globalization, outsourcing of digital design and IC fabrication has become very common, and consequently, counterfeit electronics is a major worry for application developers, especially in critical systems that involve sensing and control [6]. Such an outsourcing trend could potentially enable unauthorized devices to blend in and join the network. Thereby, authenticating devices in an IoT has become an extremely critical and challenging security threat.

An IoT is characterized by the heterogeneity of the interconnected devices, many of which are constrained in their computation, communication, and energy resources. Such resource limitation restrains the applicability of elaborate security solutions, and mandates the use of lightweight primitives and the tradeoff between security and resources [5]. In addition, IoT devices operate unattended and, thus, adversaries could come close enough to eavesdrop on transmissions [7]. In practice, to ensure secure communication, the authenticity of each device in the IoT framework should be confirmed. Accordingly, provision for efficient device authentication is highly required.

Authentication has been traditionally supported by either deploying the public-key infrastructure (PKI) [8]–[11] or identity-based encryption (IBE) [12], [13]. PKI employs one or multiple trusted parties to certify that a cryptographic key belongs to a particular user or device. Due to the associated computational and communication overhead, such certification is quite costly and not scalable for an IoT system with numerous nodes [14]. Despite their performance advantages over PKI, IBE schemes also suffer from scalability limitations and are deemed unfit for the resource-constrained IoT devices. Generally, authentication schemes that require computation-intensive cryptographic primitives, e.g., asymmetric cryptography, impose significant overhead and do not suit resource-constrained IoT devices [15]. Moreover, conventional sole-software authentication schemes [16] are not robust enough, as a device can be hacked and its cryptographic identity in terms of encryption (private) key or digital certificate can be leaked or manipulated.

Manuscript received March 10, 2021; revised May 9, 2021; accepted July 14, 2021. Date of publication July 19, 2021; date of current version February 21, 2022. *(Corresponding author: Mohamed Younis.)*

The authors are with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD 21250 USA (e-mail: e127@umbc.edu; younis@umbc.edu; nkarimi@umbc.edu).

Digital Object Identifier 10.1109/JIOT.2021.3098496

On the other hand, existing schemes that involve hardware are not secure either [12], [15]. The use of nonvolatile memories such as EEPROM or battery-backed SRAM to store shared keys are vulnerable to device tampering. The same argument applies for solutions that leverage trusted platform modules (TPMs) [17], [18]. TPM, and its lightweight alternatives [19], increase the hardware complexity and are geared for software integrity rather than device authentication. Thereby, to protect IoT frameworks, it is necessary to develop authentication and key management protocols that utilize lightweight cryptography and low-cost tamper-resistant primitives. These protocols should be versatile to be able to efficiently cope with the dynamic node membership, and resilient against contemporary attacks.

In this article, we aspire to fill the technical gap and propose a robust authentication mechanism for IoT devices. Our mechanism employs physical unclonable functions (PUFs) [20] to associate unique hardware-based IDs to the participating devices in order to enable effective protection against contemporary security threats, such as eavesdropping, impersonation, and message replay. PUFs operate based on unintentional variations that occur in the fabrication process of the integrated circuits, causing signals that follow similar paths in the design to experience slightly different propagation delays in different chips. Thereby, the response of each PUF to the same input (so-called challenge) varies among similar chips. These unique signatures are highly adopted by industry for IC Metering, detection of counterfeit ICs, and logic obfuscation [21], [22], and can also be used for device authentication purposes. Deploying PUF unique signatures alleviates the need to store the unique ID of each IoT device in memory and, thus, deprives an adversary from revealing the secure device identifier (ID) through software hacking and makes IoT devices more secure [7], [23].

A PUF is classified as weak or strong based on whether the challenge response space is small or large, respectively. Strong PUFs are often used for authentication protocols, while weak PUFs are often deemed suitable for generating cryptographic keys [4], [24]–[26]. Although PUFs are fundamentally based on random physical variations and consequently supposed to be unclonable [27], they may be prone to attacks that aim at modeling their behavior using machine learning (ML) techniques. In fact, by having access to a subset of the challenge–response pairs (CRPs), an adversary may be able to model the PUF, even strong ones [27]–[34]. Thereby, it is necessary to prevent intercepting the challenge–response exchange messages used for authenticating IoT devices.

Accordingly, this article focuses on strong PUFs and proposes a novel authentication protocol that splits challenge bit-streams into multiple messages, and engages additional (helper) nodes. The challenge bits are extracted from multiple messages routed through different nodes. The goal is to counter eavesdropping attempts aimed at uncovering the exchanged CRPs. We also study the impact of various challenge bits on the PUF modeling attack, and further provision additional protection by employing: 1) bit scrambling and 2) challenge padding techniques to degrade the adversary's modeling capabilities. Thus, this article fundamentally contributes a novel authentication protocol for IoT that: 1) employs lightweight hardware primitives; 2) avoids the reliance on cryptosystems; and 3) resists machine modeling attacks. Specifically, the contributions are as follows.

1) Develop a novel lightweight PUF-based mechanism for authenticating IoT devices. Rather than applying encryption, our mechanism pursues challenge splitting (CSP) to thwart the PUF modeling attacks.
2) Study the impact of a known portion of a challenge pattern on the PUF modeling accuracy.
3) Propose an enhancement for CSP through bit scrambling, referred to hereafter as *CSP-S*.
4) Strengthen CSP through challenge Padding (*CSP-P*) to introduce noisy data that degrades the ML-based modeling accuracy.
5) Evaluate all proposed methods using the data extracted from FPGA implementation of the target PUF.

The remainder of this article is organized as follows. Section II presents related work on IoT authentication. Section III presents the threat model considered in this study and provides some preliminaries. Section IV describes the proposed authentication mechanisms. The validation results are reported in Section V. Section VI analyzes the security and overhead of the proposed schemes. Section VII concludes this article and highlights future research directions.

## II. Related Work

An IoT is a collection of low-cost and resource-constrained devices operating in unsupervised environments [35], [36]. Even though several authentication protocols and security provisions exist for wireless networks, they do not suit the resource-constrained and very dynamic network membership of IoT devices [2], [3], [8], [9], [23]. Most existing authentication protocols rely on storing the device ID in its memory. However, such a methodology is not secure as IoT devices may not be always protected against cyber and physical attacks. To prevent storing keys in the IoT devices, deploying PUFs has been explored [27], [29], [37]–[42]. Although these authentication schemes are lightweight and benefit from the unique footprints of PUF devices, they suffer from vulnerabilities to security threats such as modeling attacks, replay attacks, and impersonation attacks.

Chatterjee *et al.* [14] used PUFs to generate public and private keys to be used for securing data transfer in IoT. The proposed scheme is resilient against replay attacks, yet it is computationally intensive and would not suit resource-constrained IoT devices. Wallrabenstein [7] opted to avoid storing the private key in the device memory in order to achieve tamper resistance. The approach is to embed an elliptic-curve cryptosystem on the IoT device, to be used along with the PUF to regenerate the private key when needed. However, the approach requires some changes to be made to the IoT device hardware. The scheme of Huth *et al.* [43] has low storage requirements; yet it needs considerable hardware changes to be applied to the device. Meanwhile, PUF-RAKE [44] uses a random number generator to shuffle the challenge and response bits and store them in an encrypted form. The selection of the random number generator is based on whether the challenge is even or odd. The device is given a sequence of random numbers to be used to reorder the provided challenge bits; upon applying the challenges to the PUF, the response bits are shuffled again before replying to the server. However, the approach either requires synchronization if the sequence of random numbers is not predetermined, or introduces vulnerability if the device can be hacked and its memory is read.

Successful modeling of the PUF behavior can compromise the PUF-based IoT security provision. Some efforts have been dedicated to mitigate the PUF vulnerability to modeling attacks. Ganji *et al.* [45]–[47] employed machine-learning techniques to model different PUF types based on their CRPs. However, those schemes need the attacker to have access to a set of CRPs that meet a specific requirement, e.g., a set of challenge bit-streams that are different in only 1 (or $n$) bits. They use the corresponding response of such a set of challenge bit-streams to determine the influential bits of the deployed PUF and increase the accuracy of the modeling attack. Our proposed schemes are resilient against such a PUF modeling attack since, by CSP, the adversary does not have access to the full challenge bits, and applying bit scrambling and padding prevents an adversary from determining the index of each challenge bit in the intercepted bit-stream.

A number of approaches have been developed to safeguard PUF-based authentication solutions against possible modeling attacks, or at least or mitigate their threat. Existing schemes can be categorized based on the methodology as: 1) hardware based; 2) encryption based; or 3) protocol based. The former opts to harness the PUF design by the incorporation of additional logic. For example, PHEMAP [48] uses a sequencer, where the challenge $C_i$ at time $t_i$ is a function of $C_0, C_1, \ldots, C_{i-1}$. Meanwhile, the objective of [49] is to increase PUF reliability and resilience to modeling. The approach is to add two flip-flops and make the output as a function of the PUF response and the first and last challenge bits. On the other hand, Gu *et al.* [30] deployed a replicated PUF, the so-called Fake PUF. Using such an extra PUF, fake CRPs are exchanged to mislead the attacker. In fact, in this method the genuine PUF is occasionally queried only at predetermined time, while the fake PUF is used frequently and is thus assumed to be queried by the adversary. Although these schemes increase the resiliency of the IoT framework against modeling attacks, they impose a significant hardware overhead, i.e., an extra PUF along with a controller and counter to decide when the fake and genuine CRPs are sent [30]. In addition, a synchronization scheme could be needed, e.g., to decide when exactly the challenge bit-stream should be sent to the genuine PUF. Our proposed CSP mechanism does not require any circuit-level modification of the basic PUF design.

Some schemes have employed a cryptosystem in order to mitigate the PUF modeling vulnerability. For example, the approach of Gope *et al.* [50] 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 [51] applies a cryptographic hash of the PUF response and stores only hashed (and encrypted) values in the database that is securely accessible by the verifier. Although the SRAM-PUF-based authentication scheme of Farha *et al.* [52] 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. CSP 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 replay and man-in-the-middle attacks.

Finally, the last category of work counters PUF-modeling through protocol-level provisions. For example, Barbareschi *et al.* [53] used predefined chains of CRPs. The authentication process fully relies on knowing the chain and only the XOR values of the responses are sent. To mitigate the vulnerability of chain leakage, multiple chains are used with disjoint links. While the PUF advantage of avoiding response storage is leveraged, chains can still be used for modeling the PUF. Some work mitigates the PUF-modeling threat by pursuing multifactor authentication, e.g., by using a shared cryptographic key in addition to the CRP [54]. Mutual authentication of IoT nodes have been tackled in [5], where the challenge bit-pattern used for authentication in a certain time slot (iteration) is determined based on the challenge that was used previously, e.g., in the preceding iteration. Also, the response is not explicitly transmitted. Such interiteration challenge dependency, along with obfuscated response transmission, degrades the adversary's ability to model the PUF. However, such a challenge selection approach makes the IoT framework vulnerable to impersonation attacks in case even one challenge is leaked. Meanwhile, Yu *et al.* [55] secured their PUF against modeling attacks via limiting the number of CRPs transferred in the IoT framework. They also prevent using repetitive challenge bit-streams to restrict launching the reliability attacks where the adversary exploits the measurement noise to model the PUF. However, such a scheme is only applicable for the cases where authentication is not conducted very often and, consequently, is not suitable for applications like self-driving cars that need to exchange data very frequently and require rapid authentication.

Overall, our CSP mechanism enables lightweight defense against modeling attacks without employing computationally intensive cryptosystems. It is worth noting that CSP also counters reliability attacks that exploit the measurement noise to model the PUF [56], since our CSP denies adversarial access to the full challenge bits (by splitting scheme) and nullifies the mapping functions [28] used by such an attack (through scrambling and/or padding schemes). We demonstrate the resilience of our proposed schemes against such an attack in Section V-B. Table I summarizes the shortcomings of the discussed state-of-the-art methods in tackling the modeling attacks.

## III. System Model and Preliminaries

### A. System and Threat Models

Our proposed solution employs hardware-based IDs for authenticating IoT devices. In particular, we assume that a PUF is embedded in each IoT device. To authenticate a device $D_i$, the server sends a request to $D_i$ that includes a challenge bit-stream. Upon receiving the request, $D_i$ will apply the challenge to its embedded PUF and send back the PUF response. By matching the node response to a preknown value, the identity of $D_i$ can be confirmed. Note that in this model, a subset of CRP combinations of each device is stored in the server during the device enrolment phase.

A PUF response could be affected by noise caused by power variation and environmental conditions, e.g., temperature [57]. Such a noise is often mitigated by the incorporation of an error correction code (ECC) at the circuit level. The focus of this article is on protocol-level protection against PUF modeling attacks using ML techniques. We are assuming that a suitable ECC, e.g., [58]–[60], is employed to ensure stability and consistency of the PUF output.

Although PUFs are supposed to be unclonable, an adversary may intercept and uncover the CRPs transferred between the server and an enrolled device. The adversary can then use the intercepted CRPs to model the behavior of the embedded PUF. In this case, the underlying device can be impersonated to introduce a wide range of malicious activities in the IoT system. Thus, it is necessary to mitigate such vulnerabilities by preventing access to CRPs. Note that the novelty of our work is in prevention of modeling attacks on PUF-based authentication schemes, rather than using PUFs for authentication.

In this article, we assume that the device authentication and enrolment management are conducted through a central supervisory node (e.g., server). The server carries out the authentication of devices either as a part of IoT admission control or as a service to enable communication between device pairs. We also assume that the server is trustworthy. Specifically, handling an IoT network with an untrusted server is out of scope of this article.

### B. Preliminaries

*1) Arbiter-PUF:* Our authentication protocol employs a strong PUF. In this article, we focus on the use of arbiter-PUFs; however, the proposed techniques can be adopted for other strong PUFs. As mentioned earlier, weak PUFs are not used for authentication and are more suitable for key generation. An arbiter-PUF is a strong PUF consisting of a pair of delay chains; when queried, it generates one response bit per challenge [20]. This PUF operates based on the process-variation that induces race between two identical paths (top and bottom paths shown in Fig. 1). The race corresponds to the difference in signal propagation delay on these two paths and affects the value latched by the arbiter [63]. The arbiter can be realized as a simple SR-latch implemented by two NOR gates. The latch output $Q$ in Fig. 1 presents the PUF ID (response). If the transition reaches the upper NOR earlier, $Q$ gets the value of "1"; otherwise, $Q$ would be "0." The value of $Q$ is important and presents the PUF ID (response). To support $L$ response bits, the circuit is replicated $L$ times, yet using the same input (challenge bits).

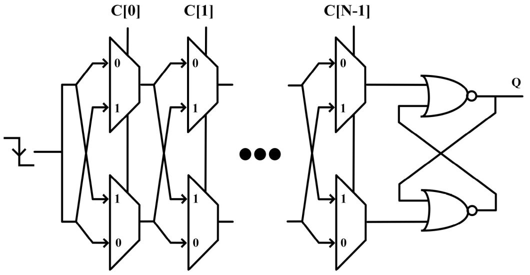


Fig. 1. Illustrating the design of an arbiter-PUF.

*2) Machine Learning:* ML is a data-driven modeling technique and is particularly effective when there is no knowledge about the process governing the data generation. The modeling performed with ML algorithms consists of two phases, namely, training and evaluation (or inference). In the training phase, the model is constructed utilizing a set of input and output data pairs. The model is adjusted based on whether it classifies the input to the correct response or not. In the evaluation phase, unseen inputs are tested to see if the model correctly determines the output.

In this article, we assume that the adversary deploys ML algorithms to model the employed PUF. In the training phase, the model is formed utilizing the PUF's CRPs. Then, in the evaluation phase, unseen inputs (i.e., challenges) are tested to see if the model correctly predicts the response. In the experiments, we use the support vector machine (SVM) [64] as well as neural networks (NNs) [65] to launch the modeling attacks.

## IV. Proposed Methodology

As mentioned earlier, to authenticate each device, the CRPs of its embedded PUF are exchanged between the server and the device. However, an eavesdropper may intercept the exchanged messages between the server and a device $D_i$ in order to uncover the CRPs; such an eavesdropper could then launch a replay attack. When sufficient CRPs are intercepted, the adversary can further develop an accurate ML-based model of the PUF to impersonate $D_i$. To mitigate such vulnerability, our approach opts to mislead the adversary about the transferred CRPs by applying the following schemes.

1) Engaging what we call "helper nodes" in the authentication process where the challenge is split among multiple packets; each packet provides only one part of the challenge bit-stream and relayed by a distinct helper node.
2) Applying a preagreed upon scrambling pattern of the whole or the subset challenge bits that are included in a packet payload.
3) Adding redundant bits to the challenge bits within the packet in a manner that is known to both server and device. Such padding further confuses the adversary about the PUF size.

Our approach does not employ a cryptosystem and is thus computationally lightweight. At the time a device is enrolled

TABLE I
COMPARISON OF THE STATE-OF-THE-ART COUNTERMEASURES AGAINST PUF MODELING ATTACKS

| Ref. | Methodology | Disadvantage |
|---|---|---|
| [5] | Mutual authentication with a sequence of dependent challenges where each challenge relates to the previous one | Vulnerable to impersonation if one challenge is leaked |
| [30] | Inserting a Fake PUF and querying it intermittently | Hardware and communication overhead |
| [44] | Challenge obfuscation by using a random number generator to shuffle the CRPs | Requiring synchronization between the node and the server for the sequence of used random numbers; also susceptibility to memory read |
| [48] | Generating challenges in a sequential manner where each challenge depends on previous challenges | Vulnerable to device impersonation attack |
| [49] | Challenge obfuscation by embedding additional logic along the arbiter-PUF | Major hardware overhead which is not suited for resource-constrained IoT applications |
| [50] | Replacing the PUF response with a pseudo response generated by a known algorithm for both device and server | Computational Overhead |
| [51] | Applying cryptographic hash function to the PUF response | High computational and hardware overhead |
| [52] | Applying cryptographic hash function to the response of SRAM PUF | SRAM PUF has a limited range of CRPs; Computational overhead; susceptibility to the man-in-the-middle attack due to using repetitive nonce |
| [53] | Authentication by XORing the PUF response with the previous responses when a predefined chain of challenges is used | Vulnerable to device impersonation attack |
| [54] | Multi-factor authentication by leveraging a shared cryptographic key in addition to the CRP | Storing symmetric key in memory defies the main advantage of PUFs and makes this scheme vulnerable to secrecy leakage |
| [55] | Limiting the number of transferred CRPs by avoiding repetitive CRPs | Only applicable when authentication is not required very often |
| [57] | Encrypting the challenge using AES for which the key is generated using a Weak PUF | Hardware overhead for implementing AES and adding an extra PUF |
| [61] | Generating each challenge based on the previous one | Synchronization among the nodes is required for correct inference of the challenge bit-stream |
| [62] | Adversarial machine learning to poison response based on the challenge bits | Can still be modeled using Neural Networks |
| [63] | Mutating the challenge and response bits using hash functions | Hardware overhead for the incorporation of two hash Functions |
| [49] | Preventing modeling attack via obfuscating the challenge bit-stream | Imposing major hardware overhead |

with the server, the protocol specifications will be determined so that the device knows which portion of the challenge bit-stream will be subject to padding and scrambling, and in what form. The idea is to synchronize the device and server while avoiding information leakage about the defense mechanism through the incorporation of any hint aboard a packet. The number of helper nodes is determined based on the network density. A helper node qualification is judged based on prior authentication or using a trust assessment/management methodology. The aforementioned three schemes are explained in detail in the balance of this section.

### A. Challenge Splitting

Unlike the usual form of transmitting challenges to an IoT node, our proposed scheme makes the server split each challenge into multiple partitions. When $D_i$ is being authenticated, it does not receive the whole challenge bits ($C_i$) from the server. Instead, the server divides the challenge bit-stream into $K$ partitions, sends one partition ($C_{i,0}$) directly to $D_i$, and arranges for the other partitions ($C_{i,1}, C_{i,2}, \ldots, C_{i,K-1}$) to reach $D_i$ indirectly through $K - 1$ other "helper" nodes. The rationale is that an eavesdropper should not be able to distinguish between the various messages to reassemble the challenge bit-stream and, correspondingly, the response of the PUF of $D_i$. We refer to this scheme as "CSP." In this scheme, the first few nodes are authenticated directly without CSP (yet via the scrambling and padding schemes discussed in the following sections); then they can serve as helpers.

In practice, the best value for $K$ can be decided based on different criteria and is subject to tradeoff. For example, the larger $K$ is, the longer the authentication delay and the more the overhead imposed on the network become. On the other hand, a large $K$ will make it more difficult for the adversary since it requires intercepting and analyzing many messages. It is noteworthy to mention that by splitting the challenge bits, first the probability of intercepting is decreased as the eavesdropper may not have access to all links due to not being within the communication range. Second, even if the adversary can eavesdrop on all links through which the PUF challenge partitions are sent, the order of partitions within the $C_i$ bit-stream will be unknown. The ordering of partitions is decided between the server and each device during the device enrolment phase in the IoT, as explained in Section IV-B.

Without loss of generality, Fig. 2 shows an example network that consists of a server and two IoT devices, $D_i$ and $D_j$, one of which is employed as a helper node. Assume that the helper node has been already authenticated and the adversary can only eavesdrop on one of the communication links, either between the server and node $D_i$ or between server and node $D_j$. Also, the response of the device can be sent directly to the server, or split and forwarded via multiple nodes. As Fig. 2 shows, the server splits the challenge into two sub-challenges with size $M$ and $N$-$M$, sends the $M$ bit portion directly to the target device and the rest through node $D_j$. In

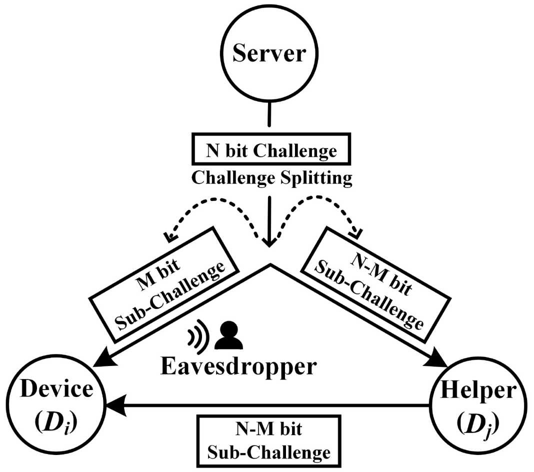


Fig. 2. Block diagram of the CSP scheme (to authenticate device $D_i$, the server uses node $D_j$ as the helper node).

practice, the helper node selection could be done dynamically based on different criteria, e.g., the proximity of the helper to the authenticated node, the presence of wireless communication links among the nodes, the time since the helper was last authenticated, etc. Helper nodes can further be qualified based on their trustworthiness that may be assessed using application-based or network-based models in the context of IoT, e.g., CTRUST [66].

Modeling the PUF with partial access to the challenge bits is very difficult, if not impossible. We will demonstrate in Section V that the success rate of the modeling attack in case of CSP is also affected by whether the most significant bits (MSBs) of the challenges are intercepted or the least significant bits (LSBs). In Section V, we will discuss how the attack success is affected based on whether the attacker knows or does not know the PUF size. We finally note that even if the challenge $C_i$ sent by the server to node $D_i$ cannot be split due to the unavailability of any trusted helper node $D_j$ (that has already been authenticated) in the communication range of $D_i$, our approach still employs bits scrambling and padding methods to boost the PUF modeling resilience. As discussed in the following sections, bit scrambling and padding can be applied even when helper nodes are involved.

### *B. Ordering of Challenge Partitions in CSP*

Challenge reformation requires knowing the order for partitions received by the device from the server directly and through helper nodes. CSP avoids explicit inclusion of control information in the packet, instead it enables the device to infer such an order based on the ID of the helper node(s) as well as the node to be authenticated. Conventionally, nodes in a given network have unique IDs that are monotonic in nature. Such monotonicity is exploited by CSP to order the received challenge partitions. Since the node ID is usually included in the packet header, an adversary can retrieve it as well. In order to prevent the adversary from concluding the partitioning order, CSP remaps the IDs using a simple hashing function similar to peer-to-peer systems. The procedure is as follows.

1) When a device $D_q$ is enrolled, the server assigns such a device a random number $\theta_q$ using a uniform distribution over the range $[U, L]$.
2) When splitting the challenge, a set of $K-1$ helper nodes is formed; assume that the IDs of the helpers to be $\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_{K-1}$. Assume that $\mathrm{ID}_K$ is the ID of $D_q$, i.e., the device to be authenticated.
3) The server calculates the rank of each involved node using: $\mathrm{Rank}_i = \mathrm{ID}_i \bmod \theta_q$
4) A sorted list $\eta$ of nodes is then formed where the device $D_q$ and helpers are sorted ascendingly based on their rank. Nodes that happen to have the same rank are sorted according to their ID.
5) The server splits the challenge into $K$ unequally sized partitions and assigns them to the nodes in $\eta$.

Device $D_q$ is to replicate the aforementioned steps since it knows $\theta_q$ and can read the helper nodes' IDs from the received packets. $\theta_q$ constitutes a secret that prevents an adversary from doing the same. We illustrate the process through an example. Assume that a device $D_q$ with ID of 103 is to be authenticated. At the time of enrollment, a random value in the range [2, 9] is picked and happens to be equal to 5, i.e., $\theta_q = 5$. The server set $K = 4$ and picked three helpers $D_x$, $D_y$, and $D_z$, whose IDs are 215, 110, and 52, respectively. Thus, the ranks of nodes $D_q$, $D_x$, $D_y$, and $D_z$ are 3, 0, 0, and 2, respectively. Since $D_x$ and $D_y$ have the same rank, we sort them according to their ID. Thus, the challenge partitions will be assigned according to the sorted set $\eta = \{D_y, D_x, D_z, D_q\}$. Upon receiving the packets from the server and nodes $D_x$, $D_y$, and $D_z$, the device extracts the challenge partitions and concatenates them according to $\eta$ in order to form the challenge bit-stream.

### *C. Challenge Scrambling*

As mentioned earlier and will be shown in Section V, in the CSP scheme, the success rate of the modeling attack increases if the MSB part of the challenge is captured, rather than the LSB part; in essence, the MSB bits are the challenge bits being applied to the multiplexers close to the arbiter as shown in Fig. 1. Accordingly, the bit position within the intercepted challenge affects the modeling attack success rate. This observation motivates our second protection scheme that performs challenge scrambling, referred to as (*CSP-S*).

In *CSP-S*, the challenge bits are reordered before being sent to the target device. For example, for an 8-bit arbiter-PUF, instead of sending the challenge bits to the target device $D_i$ as $C_i[0], C_i[1], \ldots, C_i[7]$, we can send the reordered challenge bits, e.g., $C_i[7]$, $C_i[5]$, $C_i[3]$, $C_i[1]$, $C_i[0]$, $C_i[2]$, $C_i[4]$, $C_i[6]$. In practice, all or a subset of the challenge bit-stream may be scrambled; nonetheless, the more bits are reordered, the less the attack success rate becomes. Unscrambling the challenge bits may be performed either in hardware or at the system level. The former does not impose much complexity and can be done when the scrambling scheme is fixed (static), while the latter is suitable when the scrambling algorithm changes over time. In case of fixed scrambling, the challenge bit orders should have been decided initially between the server and each device during the enrollment phase.

For the dynamic scrambling, two options are possible. The first is for the server to add some control information in the packet so that the device can know how to unscramble the challenge bits. Such an option is ruled out since the added information could be exploited by the adversary to uncover the scrambling pattern. Alternatively, the device and server agree on a similar scrambling/unscrambling algorithm that is either sequential or time-dependent in nature. The former makes the scrambling pattern in one authentication packet a function of previously used patterns, while the latter determines the scrambling pattern based on a timestamp. The algorithm could be picked during the enrolment phase. It should be noted that the inclusion of a serial number and/or a timestamp in a packet header is quite popular in order to cope with possible packet loss due to communication errors. Knowing the packet serial number and/or timestamp would not suffice for unscrambling without knowing the function. The server also varies the scrambling function across the enrolled IoT devices. The experimental results demonstrate remarkable effectiveness of *CSP-S* against modeling attacks. We also note that combining CSP and *CSP-S* can significantly increase the security of the authentication process.

### D. Challenge Padding

The objective of the *CSP-P* method is to make the authentication protocol resilient against ML-based modeling attacks by introducing extra bits in the packets that are independent from the challenge bit-stream. In *CSP-P*, we pad the challenge bit-streams with random strings. In other words, the bit-stream $C_i$ is first split into $K$ partitions $C_{i,0}, C_{i,1}, \ldots,$ and $C_{i,K-1}$; then, each partition is padded with random strings such that transmitted packets have a similar size. In essence, the adversary assumes that the padded bits are part of the challenge bit-stream and, thereby, uses them in forming the ML model. Hence, the padding bits act as noisy data and degrade the PUF modeling accuracy.

In order to enable the target device to decode the received packets and extract the underlying challenge partition, information about the challenge size and the location of the challenge bits inside the packet are preagreed upon between the server and node, i.e., defined by the server at the time a device is enrolled. In order to prevent an adversary who could guess the use of *CSP-P*, from extracting the relevant challenge bits, the format of the packet changes among devices. Fig. 3 shows two samples of such a packet payload structure; again the server varies the format among the devices for added protection in case one device is compromised.

In Fig. 3, the *Challenge Size* specifies the length of the portion of the actual challenge bit-stream included in the packet. This can notify the target device about the number of bits that it will receive through the helper node(s). Note that it is also possible to send the entire challenge bit-stream in one packet without splitting it and engaging a helper node. In that case, the device's underlying PUF does not need to wait to receive the other part of the bit-stream through helper node(s), and can evaluate the response right away. To increase the security, we do not fix the position of the challenge bit-streams inside the transferred packet, i.e., the position can change in each packet. Accordingly, the *Challenge Start Point* field informs the PUF about the starting position of the included challenge partition inside the packet. Moreover, in this method, each challenge partition is padded with two random bit-streams.

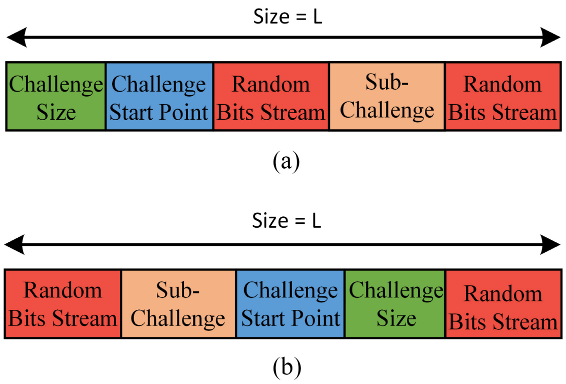


Fig. 3. Sample packet structures in *CSP-P*. Each packet includes five fields. The order and size of these fields vary from one device to another (e.g., nodes $i$ and $j$) and are picked by the server during device enrolment. The actual location of the subchallenge bits within the packet payload can change from one device to another as well as one packet to another for the same device. (a) Node $i$. (b) Node $j$.

To elaborate, let us assume that $D_i$ has an $N$-bit PUF. In this case, the packets shown in Fig. 3 designate $\lceil \mathrm{Log}_2 N \rceil$ bits for the *Challenge Size* field. Note that each authentication packet sent to $D_i$ will include $\lceil \mathrm{Log}_2 N \rceil$ bits regardless whether $D_i$ receives the challenge bit-stream in one or multiple packets. In addition, as the packet size is $L$, we assign $\lceil \mathrm{Log}_2 L \rceil$ bits for the *Challenge Start Point* field. The remaining part of the packet payload shown in Fig. 3 are used for the subchallenge bits and the random bits; the latter are added to mislead the adversary. During the enrolment phase, the server informs $D_i$ about the positions of the *Challenge Size* and *Challenge Start Point* fields in the packet; while these positions differ from one device to another, they are fixed for all packets sent to the same device. By knowing the position of these fields, the receiver can determine where the subchallenge bits "$c$" within the packet are, even if $c$ changes from packet to packet (which will be reflected in the challenge size and start points fields). Such a variability will further mislead the adversary. Again, the order and the size of each of the fields shown in Fig. 3 can change from one device to another and is determined by the server during device enrolment. Our experimental results show the efficacy of the *CSP-P* scheme.

### E. Guidelines for Protocol Selection

Fig. 4 depicts the sequence diagram of our proposed CSP authentication protocol and its variants, where $K - 1$ helper nodes are engaged in authenticating $D_i$. As will be shown in Section V, all the proposed protocols improve the resilience against modeling attacks. We expect, nonetheless, that decisions on which protocol to employ will be based on the network constraints as well as the threat model. The incorporation of padding is definitely plausible yet it elevates the bandwidth requirements and would not be attractive

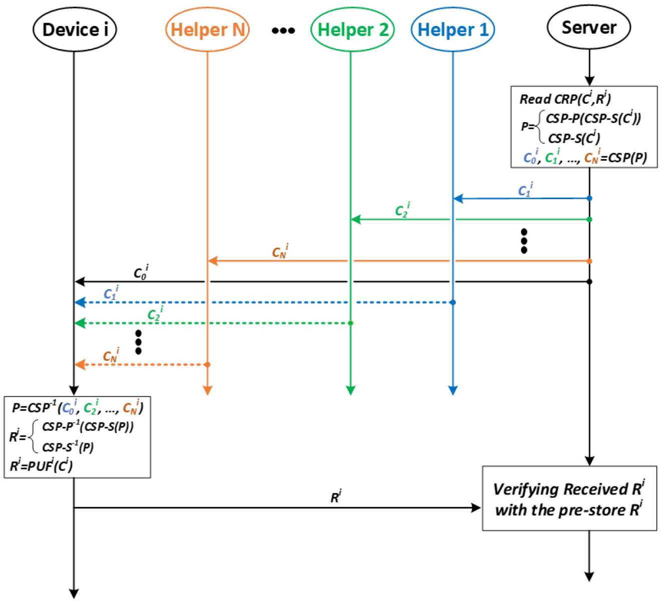

Device i
Helper N
•••
Helper 2
Helper 1
Server
Read CRP($C^i$,$R^i$)
P={ CSP-P(CSP-S($C^i$)) / CSP-S($C^i$)
$C_0^i$, $C_1^i$, ..., $C_N^i$=CSP(P)
$C_1^i$
$C_2^i$
$C_N^i$
$C_0^i$
$C_1^i$
$C_2^i$
$C_N^i$
P=$CSP^{-1}$($C_0^i$, $C_2^i$, ..., $C_N^i$)
$R^i$={ CSP-$P^{-1}$(CSP-S(P)) / CSP-$S^{-1}$(P)
$R^i$=$PUF^i$($C^i$)
$R^i$
Verifying Received $R^i$ with the pre-store $R^i$

Fig. 4. Sequence diagram for the CSP protocol with $K-1$ helper nodes being involved in authenticating the targeted IoT device $i$. The challenge $C_i$ is split by the server into $K$ partitions, where only one of these partitions is sent directly to the targeted device while the rest are sent through the helpers. In conjunction with splitting, challenge bit padding and scrambling are also employed at the level of individual packets to defend against eavesdroppers.

**Algorithm 1:** Guidelines for Applying *CSP*

```
Enrolment of IoT device $D_i$:
  • Record a set of CRPs for PUF of $D_i$
  • Assign a random value of $\theta_i$ for $D_i$
  • Agree on the structure of $CSP-P$ packet
  • Pick a $CSP-S$ scrambling algorithm
Authentication of IoT device $D_i$:
Applying $CSP-S$
if (There is a helper node in range of $D_i$) then
    Applying CSP
      1) Identifying active helper nodes in the range of $D_i$
      2) Splitting the challenge to $K$ Sub-challenges
if (The network traffic is light) then
    Applying $CSP-P$
```

when radio interference is high or in dense deployment with increased medium access contention. Scrambling can be suitable in these cases. In other words, when the packet size is a concern, we rather deploy splitting and scrambling than padding. Finally, scrambling is quite effective, yet is expected to be sensitive to how the bits are reordered. Also, if the scrambling algorithm is uncovered or modeled, the challenge bits would be recoverable by the adversary. Therefore, splitting will be invaluable as the adversary will be deprived of getting the whole challenge bits. The same can be stated regarding padding, where combining splitting and padding is a better option as the eavesdropper does not have access to some parts of the CRPs even if the padding details are leaked. Algorithm 1 provides a pseudocode summary of the required settings at the device enrollment phase and when to apply each of the CSP schemes.

## V. Experimental Results and Discussion

In this section, we first provide details of the setup used to validate the proposed schemes. Then, we present the obtained results and discuss our observations.

### A. Experimental Setup

To validate our approach, we implemented 16 and 64-bit arbiter-PUFs on Xilinx ARTIX-7 FPGA [67]. In our experiments, we dedicated one FPGA to represent the IoT node that includes an embedded PUF to be used for its authentication, and another FPGA to act as a helper node when applying our CSP scheme. The FPGA boards use the UART protocol for connecting to a PC; the latter plays the role of the server in our IoT framework. To support communication between the device and helper nodes, the two FPGAs were connected using their onboard peripheral interfaces. The PC (i.e., Server) generates a number of randomly generated bit-streams to be used as challenge bits and sends one portion to the target FPGA directly and the other portion indirectly via the helper node. These two portions are combined and used as the PUF's input challenge in the target node. The related response is sent back to the PC via the UART communication. Note that our setup is wired but it can be also implemented as a wireless infrastructure.

We employed the SVM and a NNs as representatives of ML techniques that an adversary pursues to conduct a modeling attack against the deployed PUFs. Our NN is a 5-layer fully connected architecture with one input layer (with 64 neurons reflecting the PUF size), three nonlinear hidden layers (with 5, 10, and 15 neurons), and one output neuron with sigmoid function. Rectified linear unit (ReLU) is used as an activation function in all layers. The learning rate and momentum are 0.01 and 0.99, respectively, and the number of epochs is 1000. The adversary is assumed to intercept some of the CRPs. Two scenarios are considered: 1) when the adversary intercepts a packet with the full challenge bit-streams and 2) when only some part of each challenge bit-stream is included in the intercepted packet. We note that the mapping function of [28] is used in the PUF modeling. Such a mapping reflects the structure of the arbiter-PUF considered in this article and enables successful modeling using relatively small training challenge–response sets. By using the mapping function, we are principally assuming an adversary with significant knowledge of the protection system. In these experiments, we first show the modeling results using 2000 CRPs for each PUF. Then, we increase the training size and demonstrate its impact on the launched modeling attacks when our proposed schemes are used.

### B. Experimental Results

*1) Effect of Challenge Splitting:* The first set of results assesses the resilience of CSP against modeling attacks. The results for a 16-bit PUF are shown in Fig. 5. These results were gathered for the cases in which the $N$-bit challenge ($N=16$) is split into two parts, one $M$-bit portion is sent to the node and the other $N$-$M$ bits are routed through one helper node. The assumption is that the adversary can only eavesdrop on the $M$-bit part. The bars shown in red in Fig. 5 present the

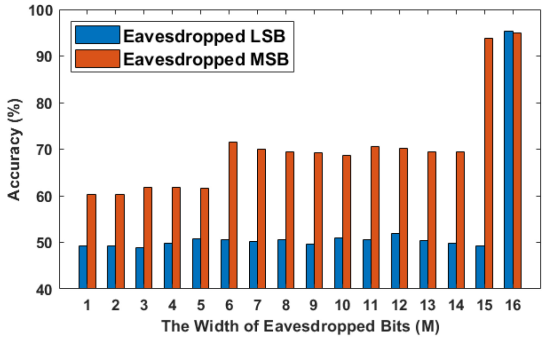


Fig. 5. PUF modeling accuracy using SVM when the adversary intercepts the $M$ LSB or MSB bits of challenge ($M \leq N$, where $N = 16$).

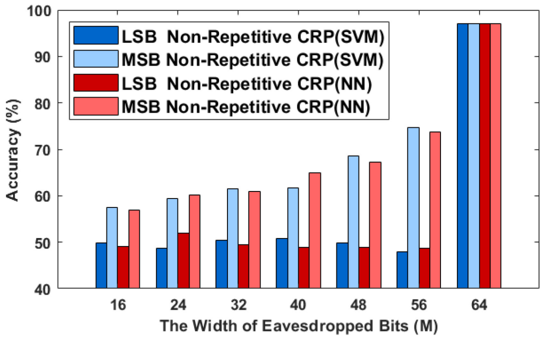


Fig. 6. PUF modeling accuracy using SVM and NN, when the CSP protocol is being applied. The adversary does not know the PUF size $N(= 64)$ and intercepts $M$ LSB or MSB bits of each challenge bit-stream.

results when $M$ MSB bits of the challenge (and intercepted by the adversary) are used to model the PUF, while the blue bars correspond to the case in which the $M$ LSB bits are used for modeling the PUF. Obviously, there is no splitting when $M$ is 16 in these experiments.

As expected, the more bits the adversary can intercept, the more accurate the PUF modeling would be. The results depicted in Fig. 5 show that getting access to the 3, 6, and 15 MSB bits results in 61.8%, 71.5%, and 93.75% modeling accuracy, respectively. A small fluctuation in accuracy (e.g., in case of intercepting 9 MSB bits) is due to the randomness of the training in ML schemes and, generally, does not affect the trend. In case of no-splitting ($M = 16$), the accuracy increases to 94.95%.

Another important trend that could be observed from the experiments is that all challenge bits do not have equivalent effects on the PUF response prediction. In other words, if the adversary could uncover $L$ (out of $N$) bits of the challenge, the accuracy of the PUF modeling significantly differs based on the position of these $L$ bits within the $N$ bit challenge pattern, e.g., $L$ MSB or LSB bits. For example, the results shown in Fig. 5 indicate that if the attacker has access to the most significant 8 bits, the accuracy is 69.40%, while by using the least significant 8 bits, the accuracy drops to 50.6%. Comparing the bars related to $M = 8$ in this figure points out the dominant effect of the MSBs in the challenge. In essence, even with access to all challenge bits except the MSB one, the adversary may not be successful in modeling the PUF, where the prediction accuracy is 49.2% for this case (corresponding to $M = 15$ in Fig. 5).

The criticality of the MSB bits of the challenge in modeling the arbiter-PUF, compared to the LSB bits, can be explained via the circuit Fig. 1. In each stage of this circuit, based on the related challenge values, either the upper and lower path inputs are connected to the related upper and lower path outputs, respectively (the so-called pass mode), or these inputs are swapped and get connected to the lower and upper path outputs, respectively (the so-called switch mode). In this case, if the attacker cannot intercept the last bit of the challenge $C[N-1]$, there is 50% probability that an incorrect value for $C[N-1]$ will be used in the ML model, even if all other challenge bits are intercepted. Such an incorrect value realizes a wrong mode, i.e., the last-level multiplexers experience the pass (switch) mode incorrectly. Thus, such an incorrect value results in 100% wrong output for that particular challenge. However, if $C[i]$ is missed by the attacker ($i < N - 1$), it is still probable that the multiplexers fed by $C[i+1], \ldots, C[N-1]$ can restore the correct output if their accumulative delay can compensate for the incorrect swap (or pass) in the stage $i$ of multiplexers. Thereby, the closer to the arbiter a challenge bit is, the more negative effect it has on the success of the modeling attack if it cannot be intercepted. Here, closer refers to the presence of fewer gates between that challenge bit and the arbiter. For example, in Fig. 1, $C[N - 1]$ ($C[0]$) is the closest (farthest) challenge bit to (from) the arbiter.

By splitting the challenge into two parts and sending one portion using a helper node, the adversary who eavesdrops on the wireless link between the node and the server may observe repetitive challenges with different responses. This may give a hint that the PUF-challenge is more than the $M$-Bit, intercepted by the adversary; otherwise, the response would not be different. The adversary also may think that the mismatch of the responses for the same challenge could be due to transmission or measurement noise; we will discuss the impact of the measurement noise later in this section. For the sake of simplicity, we have ignored the transmission noise here. However, they can be taken care of by using ECCs [27].

The results shown in Fig. 5 represent the cases in which some repetitive combinations of $M$ bits with different responses may have been encountered. To avoid redundancies, we do not show the results of modeling the 16-bit PUF when only nonrepetitive partial challenges are transferred. However, we will show the results for such a case for the 64-bit PUF later in the section. Note that the results in Fig. 5 assume that the attacker does not know the size of the embedded PUF (i.e., $N$), and guesses the size based on the partial challenge; thereby, the adversary trains the ML model based on the guessed, rather than the actual, PUF size. The case where the PUF size is known to the attacker will be discussed in the next experiments.

Fig. 6 shows the FPGA implementation results for a 64-bit PUF modeling with SVM and NN. Here, the splitting scenario is similar to the one discussed for the 16-bit PUF. This figure confirms our previous observations that if the adversary has only access to the LSB part of the challenge, regardless of the employed ML scheme, the PUF cannot be accurately modeled even with access to 48 out of the 64 bits, where the modeling accuracy is ≈ 50%. However, the trend is different when having access to the MSB part, where the accuracy grows with the increased number of intercepted challenge bits. For instance, the accuracy of 57.4%, 61.5%, and 68.5% can be achieved via access to 16, 32, and 48 nonrepetitive MSB bits, when SVM is used to model the PUF. The accuracy grows to 97% in case of no-splitting ($M = 64$). Using NN for the modeling attack results in a very similar outcome; as shown the accuracy is 57%, 61%, and 67.15% when intercepting 16, 32, and 48 nonrepetitive MSB bits when the PUF is modeled with NN.

The results shown earlier are based on the involvement of one helper node. We have also conducted experiments while *engaging two helper nodes*. When the adversary intercepts one of the 21 LSB, 21 Middle, or 22 MSB bits of the challenge, the modeling accuracy was found to be 50.35%, 51.25%, and 60.65%, respectively, while using SVM that is trained with 2000 CRPs. Using NN with the same data set gave very similar results. In these experiments the PUF size is unknown to the attacker while factoring in, rather than filtering out, repetitive CRPs. Generally, a larger helper node count makes it harder for an attacker as more links are to be monitored, as we show through analysis in Section VI.

Fig. 6 reports the performance when a set of distinct (nonrepetitive) challenge bit-streams is used to model the PUF. Repetitive challenges refer to the cases in which some of the intercepted $M$ (out of $N$) challenge bits are similar and correspond to different responses. The presence of repetitive challenge bit-streams is found not to yield noteworthy variations in the results, mainly because of the size of the challenge–response data set used in the modeling (i.e., 2000 CRPs). To better capture the effect of repetitive challenges on the performance of CSP, we rerun the experiments where SVM is applied to model the PUF using only 200 CRPs. The results shown in Fig. 7 correspond to the case where 56 MSB bits (out of 64 bits) of each challenge are intercepted by the adversary. As expected, repetitive challenges diminish the modeling accuracy. For example, having 5% repetitive challenges (out of 200) results in the accuracy of 72.15%, while with 30% and 60% repetitive challenges, the accuracy drops to 70.6% and 67.85%, respectively.

Note that the repetitive cases (similar challenges with different responses) that an adversary may observe is due to the splitting scheme. Assume that two different challenges $C_1$ and $C_2$, with different responses, are split to ($C_{1,0}$, $C_{1,1}$) and ($C_{2,0}$, $C_{2,1}$), respectively. Although $C_1$ and $C_2$ are not similar, yet $C_{1,0}$ and $C_{2,0}$ may be similar. In this case, the adversary who can only intercept the $C_{1,0}$ and $C_{2,0}$ portions as well as the PUF response is misled due to having two similar challenges with different responses. Accordingly, such data result in lower modeling accuracy.

*2) Impact of Knowing the Actual PUF Size:* In our approach, only one portion of the CRPs may be captured, specifically $M$ bits; hence, the adversary will not have access to the whole challenge to model the PUF, even with knowledge of the actual PUF size. The adversary may fill the remaining $N$-$M$ part of the challenge with random bits during training. To build the whole challenge, there are two options, namely, assuming that the available $M$ bits are the MSB or LSB parts of the challenge bit pattern, albeit if the adversary has learned about our splitting scheme.

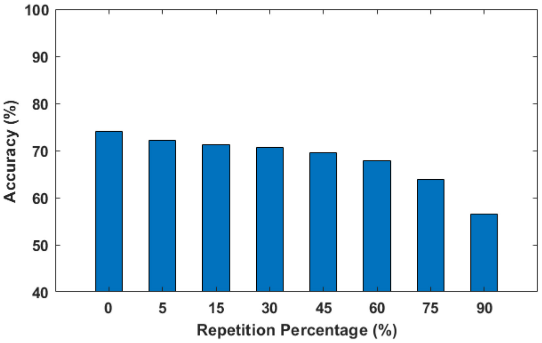
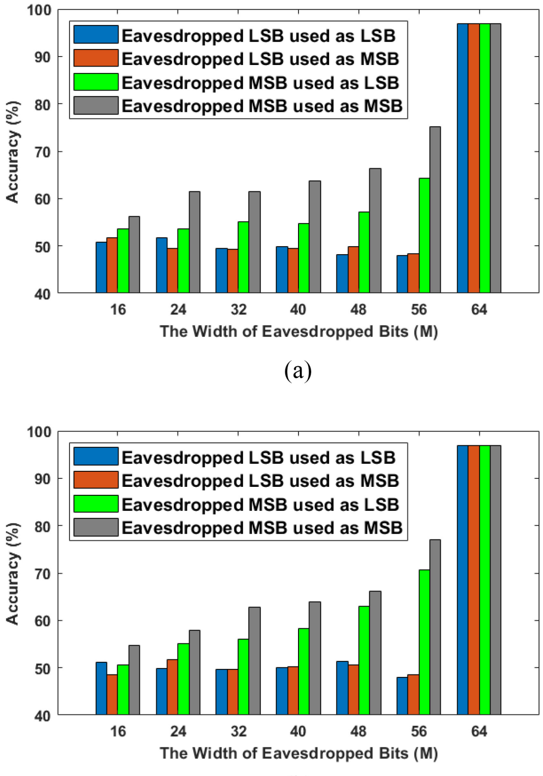


Fig. 7. PUF modeling accuracy using SVM while CSP is being applied. The adversary does not know the PUF size $N(= 64)$ and intercepts 56 MSB bits of each challenge bit-stream when $X\%$ of the intercepted challenge bit-streams are repetitive, $X \in \{0, 5, 15, 30, 45, 60, 75, 90\}$.

Fig. 8(a) and (b) depicts the results for the case where the adversary knows the PUF size, but obviously does not know whether the captured $M$ bits are the LSB or MSB portion of the original challenge. As the results indicate, the highest prediction accuracy is when the obtained bits are from MSB bits and are also treated as the MSB portion during training [shown by gray bars in Fig. 8(a) and (b)]. In this case, the accuracy is 56.15%, 61.55%, 61.5%, 63.7%, 66.4%, and 75.2% using the SVM scheme while accessing the 16, 24, 32, 40, 48, and 56 MSB bits, respectively. Meanwhile, applying NN achieves 54.8%, 57.9%, 62.85%, 63.95%, 66.1%, and 76.95% for intercepting 16, 24, 32, 40, 48, and 56 MSB bits, respectively. The first take-away point from these results is that even with using a relatively more sophisticated ML scheme, namely, NN, the attacker is not successful in modeling the PUF without having access to the full challenge bit-streams. These results are very similar to the related case in Fig. 6, where the PUF size is assumed to be unknown. Interestingly, when only the LSB is captured, learning the PUF size degrades the modeling accuracy.

*3) Efficacy of Challenge Scrambling:* In this set of experiments, bit scrambling has been applied along with splitting the challenge bits. The results reveal a significant decrease in modeling accuracy. Fig. 9 depicts the results for the case in which the challenge bits are first scrambled randomly, and then the scrambled challenge is split into two parts, where one part is sent via a helper node. These results were obtained using both SVM and NN. As shown in the figure, even if the adversary has access to the full challenge (i.e., no splitting; $M = 64$), the PUF cannot be modeled accurately, where the accuracy is ≈ 51.92% for SVM and 51.85% for NN. The take-away point from such an observation is that the challenge scrambling scheme is highly powerful in thwarting the modeling attack regardless of the ML scheme used for modeling.



(a)

(b)

Fig. 8. PUF modeling accuracy when launching attack using (a) SVM and (b) NN, while applying CSP. The adversary is assumed to know the size of PUF, and intercept $M$ LSB or MSB bits of each challenge bit-stream.

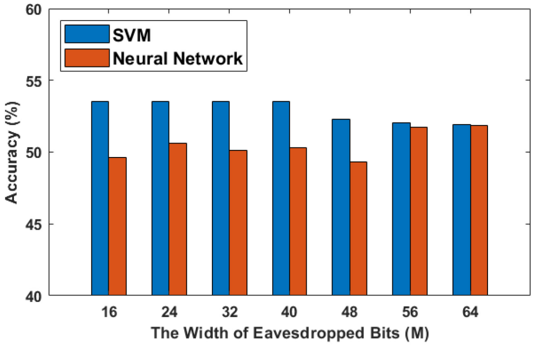


Fig. 9. Modeling accuracy using SVM and NN while the CSP-S protocol is being applied. Here, $M$ (out of 64) bits of the scrambled challenges are intercepted. Since scrambling is applied before CSP, access to MSB or LSB bits does not lead to any meaningful variations.

Note that in these experiments, for all cases of $M$, the accuracy of the modeling attack is around 50%, i.e., the slight differences observed across the bars in this figure relate to the randomness of the ML schemes and do not have much implication.

*4) Effect of Challenge Padding:* This set of results measures the efficacy of the *CSP-P* scheme in thwarting PUF modeling attacks. The results are based on the 64-bit PUF. Again, we split the challenge into two parts, and the adversary can only intercept one of them. Note that in this case, the packet size is fixed. However, the partial challenge size can be varied from one packet to another.

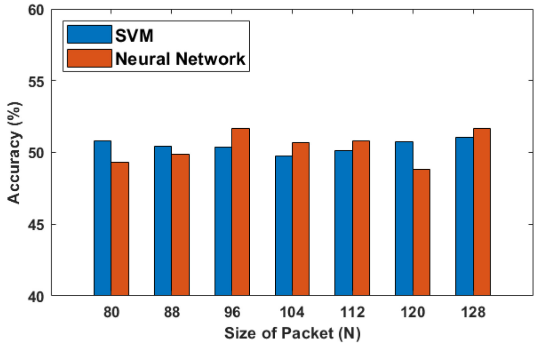


Fig. 10. Effect of the CSP-P scheme on the modeling accuracy when using SVM and NN to launch the attack.

Fig. 10 shows the accuracy of PUF modeling when SVM and NN models are applied. Each packet includes a full challenge (or part of a challenge) as well as information about the location of challenge bits within the packet payload. Based on the packet size, the payload is further padded with some random bits to mislead the adversary. For example, an 80-bit packet includes between 1 (65) and 64 (2) of challenge (padded) bits and the remaining 14 bits are devoted to specify the size of the included challenge partition along with the starting location within the packet payload (7 bits for each of "Challenge Size" and "Challenge Start Point" fields in this example as shown in Fig. 3).

When using either of SVM and NN schemes, the modeling accuracy is under 52% for all considered cases. Since the packet size is fixed during the challenge transfer, the adversary is misled and considers the packet payload size as the PUF size. Thus, the *CSP-P* scheme makes the PUF modeling attack almost impossible, where the modeling accuracy is around 50%.

*5) Effect of the Training Set Size:* For the results shown in Figs. 5–10, we used 2000 samples for training the ML model unless otherwise mentioned. In order to capture the effect of training set size on the achieved results, we have repeated the experiments for the 64-bit PUF using different numbers of training samples. Fig. 11 reports the results for the case where the adversary intercepts 32 (out of 64) challenge bits, knows the PUF size yet does not know whether the captured bits are the LSB or MSB portion of the original challenge, and uses NN for modeling. This figure shows the modeling accuracy when up to 60 000 challenge response pairs were used for training. As shown, by intercepting 50% (32 out of 64) of each challenge bit-stream, the accuracy of the modeling attack in the presence of the splitting scheme does not exceed 67%. In this case, increasing the training size beyond 20 000 samples does not result in a meaningful increase of the attack success. Note that to decrease the attack success, we can split the challenge bit-stream into more partitions, as we analyze in Section VI.

Figs. 12 and 13 capture the effect of training set size on the resilience of the scrambling and padding schemes, respectively, when all challenge bits are intercepted. As shown even with a training set of 60 000 CRPs the attacker does not have any success in modeling the PUF. These results are without

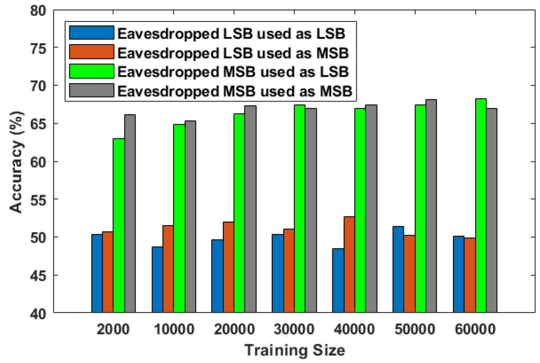


Fig. 11. PUF modeling accuracy using NN with different sizes of the training data set while CSP is applied. The adversary is assumed to know the PUF size, and intercept 32 (out of 64) challenge bits.

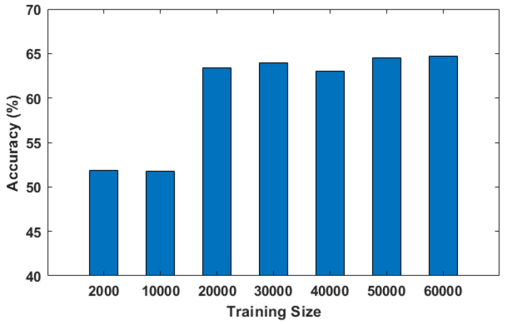


Fig. 12. PUF modeling accuracy using NN with different training data sizes when the CSP-S protocol is used. The adversary captures all 64 bits of the scrambled challenges.

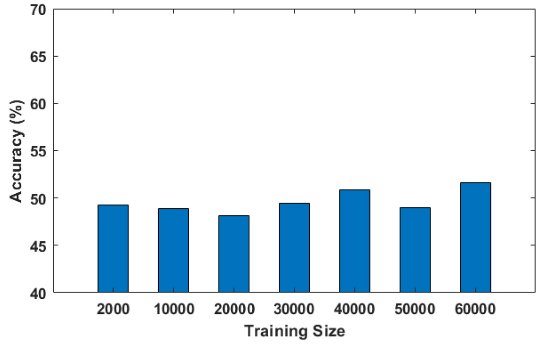


Fig. 13. Effect of the training data size on PUF modeling accuracy using NN when the CSP-P is employed. The packet size is 80 bit, and the adversary captures all challenge bits.

applying any CSP. Note that in our threat model, the adversary does not have physical access to the PUF itself and only eavesdropping on the communication links is feasible. Hence, the adversary has to monitor the links for a long time to be able to get access to 60 000 challenge response pairs; let alone being able to capture all challenge partitions and know their order when helper nodes are engaged. In summary, combining the three proposed schemes is highly effective in thwarting the modeling attacks.

*6) Resiliency Against the State-of-the-Art PUF Attacks:* To validate the resiliency of the proposed schemes, we have considered the two most prominent PUF modeling attacks.

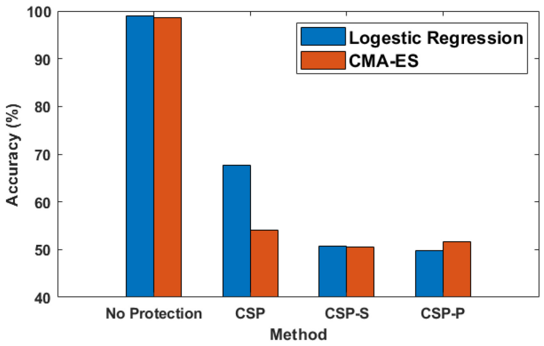


Fig. 14. PUF modeling accuracy using Logestic Regression and CMA-ES while CSP, CSP-S, and CSP-P are employed. For CSP, the adversary is assumed to know the PUF size, and intercepts 32 (out of 64) challenge bits. For CSP-S, the adversary captures all 64 bits of the scrambled challenges. For the CSP-P scheme, the packet size is 80 bit, and the adversary captures all bits. The title "No-Protection" reflects the case where none of our proposed schemes is employed.

The first is based on the logistic regression (LR) model [30], while the second is the CMA-ES attack proposed by G. T. Becker [56]. We have realized these attacks on full CRP bit-streams as well as when employing the CSP, bit scrambling and padding.

Fig 14 depicts the results when using the LR model for the attack. In these experiments, 60 000 CRPs are used for training. As shown, the accuracy of this attack is 67.75% when CSP is employed and the adversary intercepts the 32-Bit MSB part of challenge. Here, the adversary knows the PUF size yet does not know whether the captured bits are the LSB or MSB portion of the challenge. One helper node is employed in CSP; by increasing the number of helper nodes, the accuracy of modeling is expected to diminish even further as demonstrated by the security analysis in Section VI. The results of applying the LR model in the presence of CSP-S and CSP-P schemes (depicted in Fig. 14) confirm that even when all challenge-bits are intercepted, bit-scrambling and padding are highly effective in thwarting the modeling attacks; each experiment resulted in 50% modeling accuracy.

The CMA-ES-based attacks deploy the covariance matrix adaptation evolution strategy ML algorithm [68] along with reliability information obtained from the repeated measurements of CRPs. Such noise-induced reliability information is used as a side channel to assess the relative delay of the multiplexers used in the different stages of the arbiter-PUF families and, in turn, to model the behavior of the PUF. We used the open source code of the CAM-ES attack in [69] and [70] and integrated our three schemes, i.e., splitting, scrambling, and padding. The results of applying the CMA-ES attack in the presence of our protection schemes are shown in Fig. 14. In these experiments, the training set size is 60 000. As indicated by the results, in the presence of each of our proposed schemes, the accuracy does not exceed 54%. As a reference point, we also depict the accuracy of the LR and CMA-ES attacks in absence of our protection schemes. As shown, these attacks are highly successful (Accuracy ≈ 100%) when our protection schemes are not applied.

*7) Implementation Robustness and Overhead:* Uniqueness, uniformity, reliability, and randomness are important metrics based on which PUFs are evaluated [71]. The randomness is the basis for the unpredictability of PUF responses, while the uniqueness shows how well a single PUF is differentiated from other PUFs based on its CRPs. Uniformity reflects the distribution of zeros and ones in the PUF response, and reliability shows how stable the PUF response is in different environmental conditions (e.g., change in temperature). We have implemented five 64-bit arbiter-PUFs (each with an 16-bit response) in our FPGA and evaluated the randomness, uniformity, and uniqueness of each PUF via 5000 randomly chosen challenges. It has been observed that on average, the uniformity is about 49.36%, and the uniqueness among five samples is 42.24%. By increasing the number of challenges, uniqueness grew to around 50%. Both metrics ideally should be 50%.

To evaluate the reliability of the proposed architecture in different temperatures, we applied 5000 randomly generated challenges to our 64-bit arbiter PUFs 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 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 viability of our design for PUF-based authentication schemes.

The 64-bit implemented arbiter-PUFs was further evaluated using 15 statistical tests offered by NIST for assessing the randomness of true random generators [72] 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. Table II shows the results. Note that some of the tests (e.g., Universal) need larger blocks so we partitioned our responses accordingly. As shown our PUF structure passed almost all tests. This confirms the randomness of our implemented PUF.

To assess the power consumption overhead, the embedded PUF is isolated from the underlying circuit. The power consumption of a 64-bit PUF with 16 response bits was measured by the Xilinx power estimator (XPE) tool and found to be 0.002 W.

## VI. Security and Performance Analysis

There is a tradeoff between the security and the imposed overhead using the proposed methods. The overhead is fundamentally due to the increased processing and number of transmitted bits, which in turns affect power and delay. The additional processing is due to forming and decoding more packets in case of CSP or longer packets for CSP-P, and due to unscrambling in case of CSP-S. In addition, in all three schemes, building the challenge bit sequence based on the received data imposes a small delay. Moreover, there could be a little storage overhead to receive a longer packet in the case of CSP-P. In the balance of this section, we analyze the overhead and the security of the proposed schemes against PUF modeling attacks as well as conventional attacks against IoT.

TABLE II
NIST Randomness Test Result

| Test | Passed/Total | P-value |
|---|---|---|
| Frequency | 99/100 | 0.54 |
| Frequency Block | 100/100 | 0.51 |
| Runs | 98/100 | 0.50 |
| The Longest Run | 99/100 | 0.49 |
| Binary Matrix Rank | 33/33 | 0.50 |
| FFT | 99/100 | 0.50 |
| Non-overlap. Template | 99/100 | 0.51 |
| Universal | 5/5 | 0.99 |
| Linear Complexity Test | 4/4 | 0.70 |
| Serial | 98/100 | 0.49 |
| Approx. Entropy | 100/100 | 0.51 |
| Cumulative Sums | 99/100 | 0.54 |
| Random exc. | 2/2 | 0.57 |
| Random exc. var. | 2/2 | 0.40 |

### A. CSP Overhead and Resilience to Modeling Attacks

*1) CSP Resilience to Modeling Attacks:* To gauge the robustness of CSP, we analyze the difficulty of successful PUF modeling when engaging helper nodes. Assume that the server engages $K-1$ helper nodes to authenticate $D_i$. As mentioned in Section IV-A, the decision on how the full challenge bit-stream is formed at node $D_i$ after receiving all partitions is made at the time $D_i$ is enrolled in the system (before $D_i$ actually joins the network). Based on the communication range and the position of nodes, typically the adversary may be able to eavesdrop only one or a limited subset of challenge partitions. Nonetheless, we analyze the worst case scenario when all partitions are uncovered.

*Lemma 1:* When engaging $K-1$ helper nodes, the probability of capturing all individual partitions of the challenge bit-stream for a node $D_i$ is $p^K$, where $p$ is the probability of successful interception and decoding of a single packet transmission in the vicinity of the server.

*Proof:* Given the independence among the $K$ packet transmissions, the probability of intercepting the challenge packets to $D_i$ and its helpers will be $p \,.\, p^{K-1} = p^K$.

Since $p$ is a fraction, Lemma 1 implies that increasing $K$ is beneficial. For example, for an 80% packet interception probability, the engagement of two helper nodes makes the success rate for capturing all challenge partitions to be 51%. Such a rate drops to 41% when using three helper nodes. ■

*Lemma 2:* When engaging $K-1$ helper nodes and dividing the challenge into disjoint partitions, the complexity for an adversary to know the intended challenge bit-stream $C$, for a node is $K!$.

*Proof:* Let $c_i$ refers to the $i$th partition, where $C = c_1\|c_2\|\cdots\|c_K$. We consider three properties: 1) distinction, where $c_1 \neq c_2 \neq \cdots \neq c_K$; 2) asymmetry, where $c_i\|c_j \neq c_j\|c_i$ $\forall i \neq j$; and 3) nonoverlapping, where $c_i \not\subset c_j$ $\forall i \neq j$. The complexity of guessing $C$ is the highest when the distinction, asymmetry and nonoverlapping properties hold since the adversary will have to try all possible combinations for ordering the $K$ challenge partitions, for a total of $K!$ combinations.

In essence, Lemma 2 provides guidelines for comparing the various partitioning options. When any of the properties stated in Lemma 2 are violated, some of the partition ordering combinations become similar and fewer than $K!$ iterations would be needed. While it is not generally possible to achieve the distinction, asymmetry and nonoverlapping properties for all challenge bit-streams, using unequal partition sizes definitely helps. On the other hand, picking a large $K$ increases the probability that either of the three properties will be violated; thus, large $K$ increases the number of combinations yet with a trend less than $K!$. The case with maximum similarity and symmetry corresponds to when each partition is just one bit, i.e., $N$ partitions. In such a case, the number of dissimilar partition combinations is $(N!/m! \times (N-m)!)$, where $m$ is the number of "0" (or "1") bits in $C$. The analysis in the balance of this section assumes that the challenge partitions hold the properties of Lemma 2. ■

*Theorem 1:* In the worst case, the probability for uncovering the challenge bit-stream for a node $D_i$ is $(p^K/K!)$.

*Proof:* When applying CSP, the best case scenario for the adversary (worst case vulnerability) is being able to successfully find the correct challenge bit-stream. To do so, the adversary needs to: 1) intercept all challenge related packets; based on Lemma 1, such a probability is $p^K$ and 2) find the right order of the partitions by considering all possible combinations; based on Lemma 2, the probability of that is $(1/K!)$. Thus, the probability of the worst case scenario is $(p^K/K!)$. ■

*Theorem 2:* In the worst case, the runtime complexity of launching a successful modeling attack against a node $D_i$ when CSP is applied is $\mu K!$, where $\mu$ is the average runtime complexity of the underlying ML scheme.

*Proof:* The worst case vulnerability for CSP is when the adversary successfully uncovers all $K$ challenge partitions. In such a case, the adversary will have to try all possible partition orderings and for each a ML model has to be established. Based on Lemma 2, the adversary will have to form $K!$ distinct ML models and, thus, the runtime complexity is $\mu K!$.

Based on Theorem 1, even if the adversary has access to all the challenge partitions, by not knowing how to sort them out the probability of successful PUF modeling is quite low. Noting that $p$ is a fraction, the probability of a successful attack in fact exponentially diminishes with increasing $K$, i.e., the number of helpers. Similarly, the runtime complexity is prohibitive and grows with $K$, as indicated by Theorem 2. Finally, we stress that missing some of the challenge partitions will hinder the modeling process all together as demonstrated by the results in Section V. ■

*2) CSP-Related Overhead:* Increasing resilience to attacks comes at a price of increased overhead. Here, we analyze the overhead imposed by our CSP scheme. The more helper nodes are involved in the process of sending a challenge bit-stream, the higher the traffic overhead and, in turn, the total energy consumption, becomes. To formulate the traffic (or energy) overhead, assume that each packet $i$ consists of $H_i$ bits header and $W_i$ bits data. The header size ($H_i$) is constant for all packets (referred to as $H$ hereafter) while the length of $W_i$ varies based on the number of challenge bits the packet includes. In an IoT framework with an $N$-bit PUF embedded in each IoT device, $N+H$ bits are transferred per challenge. However, when CSP engages $K-1$ helper nodes, the total number of transferred bits is shown in (1), where the first term relates to the bits transferred between the server and the helper nodes (including node $D_i$ itself) and the second term shows the number of bits transferred between the $K-1$ helper nodes and the target device, i.e., $D_i$

$$\begin{aligned}\text{Total \# of Bits} &= \sum_{i=0}^{K-1}(H+W_i) + \sum_{i=1}^{K-1}(H+W_i)\\ &= (2K-1)H + W_0 + 2\sum_{i=1}^{K-1} W_i. \qquad (1)\end{aligned}$$

Equation (2) represents the case, where the $N$-bit challenge bit-stream is divided into $K$ equally sized partitions. As shown, the greater the number of helper nodes is, the higher the overhead becomes. However, note that with involving more helper nodes, the probability that the adversary can successfully eavesdrop on multiple channels diminishes and, thus, the system is more secure as confirmed by Theorems 1 and 2, above

$$\begin{aligned}W_i &= \frac{N}{K} \quad i \in \{0, 1, 2, \ldots, K-1\}\\ \text{Total \# of Bits} &= (2K-1)H + \frac{N}{K} + 2\sum_{i=1}^{K-1}\frac{N}{K}\\ &= (2K-1)\left(H + \frac{N}{K}\right)\\ &= (2K-1)H + \left(2N - \frac{N}{K}\right). \qquad (2)\end{aligned}$$

### B. CSP-S Security and Overhead Analysis

*1) CSP-S Resilience to Modeling Attacks:* To analyze the resiliency of CSP-S against modeling attacks, we recall that the order of bits in a challenge bit-stream is highly influential for predicting the response of some PUF-types, e.g., the arbiter-PUF family considered in this article. Here, we focus on the scenario when the attacker intercepts all challenge bits and opts to overcome the bit scrambling scheme by trying all possible combinations (i.e., brute force). We note that when scrambling is combined with CSP, the modeling attack complexity will substantially grow since the aforementioned analytical results would apply as well.

*Lemma 3:* Fixed (static) scrambling of the challenge bits degrades the PUF modeling attack if a PUF-design mapping function is used.

*Proof:* Modeling the PUF fundamentally opts to determine a function $f: N \rightarrow 1$ for each bit in the PUF response, where $N$ is the size of the challenge bit-stream. If the design of the underlying PUF, e.g., arbiter, is not factored in, fixed scrambling will simply yield a consistent style of bit reshuffling and will not impact the ML scheme. However, considering the PUF design, may enable modeling it via using significantly smaller training data. For such a case, fixed scrambling will disturb the design mapping function and diminish the accuracy of the PUF model for the same training data set.

It is noteworthy that the use of the mapping function of [28] to facilitate the modeling of the arbiter PUF is quite common; hence, the attacker's application of such a mapping function is expected. For the sake of comparison, we have studied the modeling of a 64-bit PUF using NNs with and without exploiting the mapping function. The results show that a modeling accuracy of $\approx 97\%$ could be achieved with as little as 2000 challenges when the mapping function is taken into account; without the mapping function the accuracy is 52% even with using 2 000 000 CRPs. Hence, without the mapping function the PUF modeling is ineffective regardless whether scrambling is used or not. Nonetheless, CSP degrades the modeling attack as confirmed by Lemma 3. ■

*Lemma 4:* Dynamic (varying) scrambling of the PUF challenge bits boosts the complexity of the modeling attack by a factor of $(N!/[m! \times (N-m)!])^{|S|}$, where $N$ is the size of the challenge bit pattern, $S$ is the set of CRPs used for training the ML scheme, and $m$ is the average number of "0" (or "1") bits in $C \in S$.

*Proof:* Assume that the average runtime complexity of the underlying ML technique is $\mu$. Scrambling the $N$ bits of the individual challenges using inconsistent patterns, e.g., time varying patterns, will necessitate the consideration of all possible ordering options $\Psi$, which has been shown earlier to be $(N!/m! \times (N-m)!)$ for a challenge with $m$ zero bits. Thus using a training data set of size $S$ requires the adversary to consider $\Psi^{|S|}$ different combinations of challenges taking into account that each challenge bit-stream may have been scrambled in a different way (using the dynamic scheduling scheme). Building an ML model for each possible option results in $\Psi^{|S|}$ trials in the worst case. This implies elevating the runtime complexity of the modeling attack to $\mu\Psi^{|S|}$. ■

*Theorem 3:* When employing CSP-S with $K-1$ helper nodes, the probability of successful PUF modeling attack is $(p^K/\Psi^{|S|})$ for time-variant scheduling, where $N$ is the PUF size, and $S$ is the size of database used for training, and $\Psi = (N!/[m! \times (N-m)!])$ with $m$ being the average number of "0" (or "1") bits in $C \in S$.

*Proof:* Based on Lemma 4, the adversary has to consider all possible $\Psi^{|S|}$ bit orderings. Thus, the probability of having the correct bit pattern is $(1/\Psi^{|S|})$. A successful attack will be the conditional probability of having the right challenge pattern given the interception of all $K$ challenge packets, which has the probability of $p^K$. Assuming statistical independence, the overall probability of a successful modeling attack is the product and is thus $(p^K/\Psi^{|S|})$. ■

*2) CSP-S Overhead:* The overhead imposed by bit scrambling depends on whether fixed or dynamically changing patterns are being pursued. Applying a fixed pattern does not impose any processing or transmission overhead since the pattern does not vary after both the server and device agree on during device enrollment. On the other hand, dynamic scrambling could impose some processing overhead. As stated in Section IV-C, in the dynamic case, the scrambling function can be sequential or time-dependent using the timestamp and/or the sequence number in the authentication packet header. Again, the inclusion of a timestamp and a sequence number is quite conventional in practice in order to detect packet loss and, hence, would not constitute an overhead for CSP-S.

### C. CSP-P Overhead and Modeling Attack Resilience

*1) CSP-P Resilience to Modeling Attacks:* Recall that padding adds a few extra bits to the payload of the challenge packet in order to mislead the adversary about the real size of PUF as well as which bits among the packet payload relates to the challenge. Let us assume that for padding and the associated control bits a total of $E$ extra bits are added to packet payload.

*Lemma 5:* In the worst case, the probability of uncovering a challenge bit-stream for a device $D_i$ that is applying CSP-P is $1/(E+1)$ when the PUF size, $N$, is known to the adversary.

*Proof:* When the adversary does not know the size of PUF, all $N+E$ bits will be used for modeling. As in our method, the padding is dynamically changed, i.e., the place of the $N$ bit challenge may change in the $N+E$ bit-stream, PUF modeling would be highly difficult, if not impossible. However, in the case of knowing the PUF size, the adversary has to select $N$ consecutive bits from the $N+E$ bits packet payload. In that case $E+1$ possible combinations have to be tried for each challenge bit-stream, i.e., the probability of uncovering each challenge is $1/(E+1)$. We assume that the adversary knows how to distinguish between the payload and packet header and can extract the $N+E$ from the intercepted packet. ■

*Theorem 4:* When combining CSP-P with scrambling, the probability of revealing each challenge bit-stream $C$ is $1/((E+1) \times \Psi)$, where $N$ is the PUF size, and $\Psi = N!/(m! \times (N-m)!)$ with $m$ being the average number of zeros in $C$.

*Proof:* As mentioned in Lemma 5, the probability of revealing the challenge bit-stream in CSP-P is $(1/E+1)$. When the challenge bit-stream is scrambled before padding, based on Lemma 4 there may result in $\Psi$ different combinations. Thereby, if CSP-P is applied to such a scrambled bit-stream, the probability of uncovering the challenge would be $[1/((E+1) \times \Psi)]$. ■

*2) CSP-P Overhead:* When CSP-P is applied, instead of transferring $H+N$ bits for sending each challenge bit-stream to node $D_i$, $H+N+E$ bits are sent. The bigger $E$ gets, the larger the overhead becomes, yet the higher security of the system is. If padding is combined with our CSP scheme, i.e., sending the padded challenge bit-stream using $K-1$ helper nodes, the overhead can be computed by using (1), where $W_i$ includes the challenge bits along with padding bits ($E_i$) in each packet $i$. If the $N$-bit challenge is divided into $K$ equally sized partitions, the same number of padding bits ($E$) is needed for each packet, consequently the total number of bits exchanged to send an $N$-bit challenge bit-stream can be estimated based on (2) to be: $(2K-1) \times (H+(N/K)+E)$. In case of combining scrambling with padding, the former does not impose any extra overhead.

### D. Resiliency Against Conventional Attacks

*1) Defeating the Splitting Scheme:* Splitting the challenge bit-stream makes it almost impossible for an attacker to collect CRPs for an IoT device (say $D_i$), even with intercepting all inbound packets. Basically, the adversary cannot determine whether a packet is intended for $D_i$ or $D_i$ acts as a helper node. In addition, to rebuild the full challenge from its portion, the adversary should know the splitting algorithm (recall the effects of MSB and LSB portions).

*2) Preventing Replay Attacks:* As mentioned above, there is little possibility that an adversary can rebuild a full challenge from its portions even with intercepting all the incoming packets to the node that is being authenticated ($D_i$). Accordingly, our approach prevents replaying a response packet from $D_i$.

*3) Countering Impersonation Attacks:* IoT frameworks are vulnerable to impersonation attacks, where a malicious node claims the identity of a legitimate one by eavesdropping on the communication traffic and replaying authentication messages. However, our approach counters such an attack, as even when the server uses the same challenge for authenticating a specific node, the packet is split dynamically to two packets (or more in case of multiple helper nodes), each of which is potentially sent via a different route. Thus, to conduct impersonation, the adversary not only has to eavesdrop on all routing paths but also needs to know the splitting algorithm, which is almost impossible without excessive resources, as shown earlier in this section.

### E. Effect of Eavesdropping Range

Modeling the PUF requires the adversary to capture a sufficiently large number of CRPs in order for the employed ML technique to yield high accuracy. In our system model, we assume that the adversary eavesdrops on the targeted device to intercept the transmissions from the server and extract the exchanged CRPs. CSP counters such an attack by splitting the challenge bits among different packets that are routed to the targeted device through helper nodes, and employing bit scrambling and padding. Here, we direct our attention to the interception range of the adversary, particularly what happens if the adversary can eavesdrop on multiple nodes. This issue is related to the node density, the underlying wireless transmission technology, and the employed communication protocols. For example, WiFi supports ranges of up to 92 m, which enables an adversary to capture packets sent by the server to quite a few nodes, some of which may be playing the role of helpers during device authentication. Analyzing these packets collectively could be pursued by the adversary in order to infer the operation of CSP and uncover the challenge response pairs. Such a concern grows in scope with increased node density since the probability of having both the device and its helpers within the interception range increases. The underlying networking protocol could further assist the adversary by embedding IDs in the packet header that distinguishes among packet receivers.

Nonetheless, assuming an attacker intercepts all packets related to the CSP protocol, analyzing these packets requires trying all combinations (i.e., brute force) causing the runtime complexity to be exponential, as we have shown earlier in this section. By appropriately setting the various parameters, the system designer can diminish the risk of such an attack scenario. For example, as indicated by Theorem 4, engaging multiple helpers and employing a large PUF would massively degrade the attack success probability. In Section V, we have demonstrated that modeling the PUF using a subset of the challenge bit-stream will not be beneficial either. Moreover, applying anti-traffic analysis measures, e.g., the use of time varying pseudonyms in the packet headers, will mitigate the threat of packet correlation and identifying helper nodes. With that said, if the details of the CSP configuration is discovered and the attacker can intercept all traffic and infer relationships between nodes, i.e., identify helpers of a device, the attacker could eventually uncover the CRPs. However, we deem such a scenario to be very improbable with appropriate CSP parameter settings and employing contemporary traffic analysis countermeasures. The latter is a well-studied topic and is out of the scope of this article.

### F. Comparing CSP With Conventional Cryptosystems

Here, we compare our CSP protocol with the alternative approach of using packet encryption to protect the challenge bit-stream. We consider the conventional symmetric and asymmetric cryptosystems and compare the performance in terms of energy consumption and delay for transferring the data between server and the IoT device to be authenticated. To have a baseline for our comparison, we consider the Jennic JN5139 communication model [73], which employs an IEEE 802.15.4/ZigBee transceiver that operates on 2.3–3.6 V and has output power of 2.5 dbm. Considering the typical proximity among IoT nodes, we can assume an output power of 1 dbm, which corresponds to ≈ 1.2 mW. The following discussion shows the estimation for the energy consumption of IoT nodes.

*Energy:* In the JN5139 module, the drawn current during data transmission ($T_x$) and reception ($R_x$) are 15 and 17.5 mA, respectively. By assuming a supply voltage of 2.9 V

$$\begin{aligned} T_x \text{ Power} &= (2.9 \times 15) + 1.2 = 44.7 \text{ mW} \\ R_x \text{ Power} &= (2.9 \times 17.5) = 50.75 \text{ mW}. \end{aligned} \tag{3}$$

The maximum raw data throughput for the IEEE 802.15.4/ZigBee transceiver is 250k bits per second; hence

$$\begin{aligned} \text{Energy per } T_x \text{ Bit} &= \frac{44.7 \text{ mW}}{250,000 \text{ bit/s}} \approx 179 \text{ nJ/bit} \\ \text{Energy per } R_x \text{ Bit} &= \frac{50.75 \text{ mW}}{250,000 \text{ bit/s}} \approx 203 \text{ nJ/bit}. \end{aligned} \tag{4}$$

As an example, let us consider the case where a 64-bit PUF is embedded in each IoT device and CPS employs three helper nodes during the authentication. By splitting the challenge equally among the device and helpers, each CPS packet will have 2-byte payload. Assuming a 4-byte packet header, the energy per transmitted and received CPS packet would be

$$\begin{aligned}\text{Energy}/T_x \text{ packet} &= \frac{179 \times 8 \times (4+2)}{1000} \approx 0.009 \text{ mJ}\\ \text{Energy}/R_x \text{ packet} &= \frac{203 \times 8 \times (4+2)}{1000} \approx 0.01 \text{ mJ}. \quad (5)\end{aligned}$$

During authentication, the IoT device will receive 4 packets with transmission energy of 0.04 mJ (0.01 mJ for each), and each helper node receives and sends one packet with a total energy overhead of 0.019 mJ. Therefore, the overall consumed energy is $3 \times 0.019 + 0.04 \approx 0.1$ mJ.

Rather than using CSP with plain text, let us assume that encryption is used. Kim *et al.* [74] have compared the energy consumed by asymmetric and symmetric encryption algorithms. Specifically, they have considered an elliptic curve integrated encryption scheme (ECIES) for private–public key encryption and the advanced encryption standard (AES) algorithm for symmetric encryption. To suit the resource-constrained devices such as the Jennic JN5139 module, small key sizes, specifically, 256 and 128 bits, were picked for ECIES and AES, respectively. The results have shown that ECIES consumed 1230 times and 250 times more energy than AES-128 during encryption and decryption, respectively. Meanwhile, according to the infamous BearSSL library [75], cryptographic hash has close execution time to AES and, consequently, they have similar energy consumption profile. Hence, it is sufficient to focus only on AES in our analysis.

Assuming a 128-bit key, an encrypted packet will have a 128-bit payload and 4 Bytes header; thus, the device will consume $203 \times (128 + 32) = 32,480$ nJ $\approx 0.032$ mJ in communication. A recent study of various implementations of AES on IoT devices has shown that the energy consumed in applying AES decryption is in range of 5–34 mJ, depending on the implementation [76]. In other words, the use of a lightweight cryptosystem imposes at least 5.032 mJ ($5 + 0.032 = 5.032$ mJ), on the device to retrieve the challenge sent by a server. Note that in CSP, the energy would be around 0.04 mJ for receiving the challenge packets since only simple operations, such as basic bit truncation and concatenation, are needed for the modulo operation and challenge reconstruction from the received packets. Even when considering the overall energy consumed by all involved nodes collectively, the total energy overhead stands at 0.1 mJ (as computed earlier), which is still insignificant compared to the case of AES. The gap between CSP and an AES-based implementation is so wide that the superiority of our approach holds even if the AES energy consumption is significantly reduced.

Similarly, deploying lightweight LFSR-based stream ciphers such as Trivium to encrypt the challenge bits before transmission is not appropriate considering their energy consumption. As reported in [77], Trivium consumes around 81 mJ on a single-board microcontroller IoT platform, which is still very high compared to CSP. It is noteworthy to mention that using single LFSR, instead of the multiple LFSRs deployed by Trivium, is not recommended as it can be vulnerable to attacks [78].

*Latency:* Fundamentally, our CSP protocol splits the challenge bit-stream among $K$ packets and does not embed any additional control information. Hence, the delay overhead is mainly due to sending $(K-1)$ packet headers corresponding to the helper nodes. Assuming $\Delta$ is the time for sending a packet header, the delay overhead equals $(K-1)\Delta$. The alternative to our approach is to use packet encryption, where the delay is due to: 1) the increased packet load since the encryption key is usually longer than the challenge bit-stream and 2) the relatively long execution time at the device to decrypt the packet and retrieve the challenge bits. Particularly, the latter typically dominates (given the limited computational capacity of IoT devices) and makes the CSP delay overhead to be insignificant compared to the use of packet encryption.

TABLE III
COMPARING CSP WITH CONVENTIONAL CRYPTOSYSTEMS

| Authentication Method | Energy Consumption (mJ) | Authentication Time (mS) |
|---|---|---|
| CSP-Based Authentication | 0.1 | 0.768 |
| AES-Based Authentication [76] | 5.03 | 29.24 |
| LFSR-Based Authentication [77], [79] | 81 | 2.8 |

In order to further illustrate the superiority of CSP in terms of latency, we compare the delay imposed when AES is used to secure the transmitted challenges with the case that CSP is employed. We again consider the time for sending a challenge partition of 2 bytes along with a 4-byte packet header (as discussed earlier), which requires $48/250\,000 = 0.192$ ms. Hence, it will take 0.768 ms for a device to receive all four partitions. Reassembling the challenge is through simple concatenation operation and would be in the nano seconds range. Meanwhile, Tsao *et al.* [76] has measured the execution time of AES with a 128-bit block size on a Raspberry Pi-based IoT platform and reported that it takes between 28.6 and 108.5 ms to decrypt a message depending on the AES algorithm implementation. As an encrypted challenge packet by AES will have a 128-bit payload, by assuming a 4-Byte packet header, the encrypted challenge packet will need($[128 + (4 \times 8)]/250\,000) = 0.64$ ms to be transmitted over a ZigBee link. Thus, in the baseline case, where the challenge is sent in an encrypted form, it will take a device at least $28.6 + 0.64 = 29.24$ ms to retrieve the challenge. Obviously, our CSP protocol is very advantageous by protecting the challenge without the use of a cryptosystem.

Finally, a recent study has reported the execution time of popular lightweight LFSR-based stream ciphers for IoT, such as Lizard, Fruit, Plantlet, and Espresso [79]; the latter is developed for 5G systems. The study is conducted by implementing these ciphers on an Arduino platform that has a 16-MHz ATmega 328P microcontroller and 32-kB RAM. The reported results indicate that these ciphers take around 76 ms to encrypt 256 Bytes. Assuming the best case scenario that there is no setup time for the stream cipher and that the execution time is just proportional to the data size, it would take about 2.4 ms for a 64-bit PUF challenge. The packet will consist of 64-bit payload and 4 bytes overhead and consequently will take $([64+(4\times 8)]/250\,000) = 0.38$ ms to be transmitted. Thus, the latency for any of the aforementioned LFSR-based techniques would be $2.4 + 0.38 \approx 2.8$ ms, which is about four times worse than the latency for CSP. If we factor in the cipher setup time, the performance advantage of our methods over stream ciphers will just grow in significance. Table III summarizes the discussed comparison.

## VII. Conclusion

In this article, we have developed an effective and lightweight PUF-based authentication protocol for IoT devices. The protocol employs three novel schemes, namely, CSP, scrambling, and padding, that hinder the adversary's ability in retrieving the challenge bits of the PUF without reliance on cryptosystems. Along with introducing variability in the packet format and not embedding any control information, the proposed schemes achieve resiliency against an adversary that intercepts the exchanged packets and opts to model the PUF behaviors using ML techniques. Through analysis and simulation, we have shown that engaging helper nodes to exchange the embedded PUFs' signatures, makes the modeling attacks very difficult. Moreover, the validation results have confirmed that via scrambling and/or padding the exchanged PUF challenge the success of the modeling attack diminishes further. As future work, we plan to develop PUF-based data integrity solutions and devise the associated key management protocols.

## References

[1] R. Taylor, D. Baron, and D. Schmidt, "The world in 2025—Predictions for the next ten years," in *Proc. IMPACT*, 2015, pp. 192–195.

[2] T. A. Ahanger and A. Aljumah, "Internet of Things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2019.

[3] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.

[4] T. Idriss *et al.*, "A PUF-based paradigm for IoT security," in *Proc. World Forum Internet Things (WF-IoT)*, 2016, pp. 700–705.

[5] M. N. Aman, K. C. Chua, and B. Sikdar, "Position Paper: Physical unclonable functions for IoT security," in *Proc. Int. Workshop IoT Privacy Trust Security*, 2016, pp. 10–13.

[6] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[7] J. R. Wallrabenstein, "Practical and secure IoT device authentication using physical unclonable functions," in *Proc. FiCloud*, 2016, pp. 99–106.

[8] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *Proc. ICCAD*, 2014, pp. 417–423.

[9] X. Liu, M. Zhao, S. Li, F. Zhang, and W. Trappe, "A security framework for the Internet of Things in the future Internet architecture," *Future Internet*, vol. 9, no. 3, p. 27, Jun. 2017.

[10] X.-W. Wu, E.-H. Yang, and J. Wang, "Lightweight security protocols for the Internet of Things," in *Proc. IEEE PIMRC*, 2017, pp. 1–7.

[11] N. Hong, "A security framework for the Internet of Things based on public key infrastructure," in *Advanced Materials Research*, vol. 671. Cambridge, MA, USA: Trans. Tech. Publ., 2013, pp. 3223–3226.

[12] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[13] S. W. Jung and S. Jung, "Personal oauth authorization server and push oauth for Internet of Things," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 6, p. 16, 2017.

[14] U. Chatterjee *et al.*, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Trans. Depend. Secure Comput.*, vol. 16, no. 3, pp. 424–437, May/Jun. 2019.

[15] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the Internet of Things," in *Proc. Int. Conf. Future Netw. Distrib. Syst.*, 2017, pp. 1–8.

[16] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proc. IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016.

[17] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust," in *Proc. NDSS*, vol. 12, 2012, pp. 1–15.

[18] G. Dessouky, T. Abera, A. Ibrahim, and A.-R. Sadeghi, "LiteHAX: Lightweight hardware-assisted attestation of program execution," in *Proc. ICCAD*, 2018, pp. 1–8.

[19] C. Shepherd *et al.*, "Secure and trusted execution: Past, present, and future—A critical review in the context of the Internet of Things and cyber-physical systems," in *Proc. Trustcom/BigDataSE/ISPA*, 2016, pp. 168–177.

[20] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. DAC*, 2007, pp. 9–14.

[21] *Physically Unclonable Function*. Accessed: Mar. 2021. [Online]. Available: https://www.secure-ic.com/solutions/security-ips/physically-unclonable-function/

[22] *What Makes PUF Technology One of the Best Protections in Cryptography?* Accessed: Mar. 2021. [Online]. Available: https://www.maximintegrated.com/en/design/blog/what-makes-puf-technology-one-of-the-best-protections-in-cryptography.html

[23] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for Internet of Things," *Comput. Netw.*, vol. 183, Dec. 2020, Art. no. 107593.

[24] O. Günlü, "Multi-entity and multi-enrollment key agreement with correlated noise," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1190–1202, 2021.

[25] L. Kusters and F. M. J. Willems, "Secret-key capacity regions for multiple enrollments with an SRAM-PUF," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2276–2287, Sep. 2019.

[26] H. Yıldız, M. Cenk, and E. Onur, "PLGAKD: A PUF-based lightweight group authentication and key distribution protocol," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5682–5696, Apr. 2021.

[27] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 3, p. 67, 2017.

[28] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. CCS*, 2010, pp. 237–249.

[29] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," in *Proc. DAC*, 2014, pp. 1–6.

[30] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing PUF based authentication," in *Proc. AsianHOST*, 2019, pp. 1–6.

[31] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-puf based authentication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1183–1196, Jun. 2021.

[32] M. Khalafalla and C. Gebotys, "PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," in *Proc. DATE*, 2019, pp. 204–209.

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

[34] J. Delvaux, "Machine-learning attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2043–2058, Aug. 2019.

[35] R.-H. Hsu, J. Lee, T. Q. S. Quek, and J.-C. Chen, "Reconfigurable security: Edge-computing-based framework for IoT," *IEEE Netw.*, vol. 32, no. 5, pp. 92–99, Sep./Oct. 2018.

[36] M. N. Aman, K. C. Chua, and B. Sikdar, "Secure data provenance for the Internet of Things," in *Proc. Int. Workshop IoT Privacy Trust Security*, 2017, pp. 11–14.

[37] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proc. S&P*, 2012, pp. 33–44.

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

[39] S. Schulz, A. Schaller, F. Kohnhäuser, and S. Katzenbeisser, "Boot attestation: Secure remote reporting with off-the-shelf IoT sensors," in *Proc. ESORICS*, 2017, pp. 437–455.

[40] Y. Lao, B. Yuan, C. H. Kim, and K. K. Parhi, "Reliable PUF-based local authentication with self-correction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 2, pp. 201–213, Feb. 2017.

[41] M. Barbareschi, P. Bagnasco, and A. Mazzeo, “Authenticating IoT devices with physically unclonable functions models,” in *Proc. 3PGCIC*, 2015, pp. 563–567.

[42] M. N. Aman, K. C. Chua, and B. Sikdar, “Mutual authentication in IoT systems using physical unclonable functions,” *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.

[43] C. Huth, J. Zibuschka, P. Duplys, and T. Güneysu, “Securing systems on the Internet of Things via physical properties of devices and communications,” in *Proc. IEEE Syst. Conf. (SysCon)*, 2015, pp. 8–13.

[44] M. A. Qureshi and A. Munir, “PUF-RAKE: A PUF-based robust and lightweight authentication and key establishment protocol,” *IEEE Trans. Depend. Secure Comput.*, early access, Feb. 10, 2021, doi: 10.1109/TDSC.2021.3059454.

[45] F. Ganji, D. Forte, and J.-P. Seifert, “PUFmeter a property testing tool for assessing the robustness of physically unclonable functions to machine learning attacks,” *IEEE Access*, vol. 7, pp. 122513–122521, 2019.

[46] F. Ganji, D. Forte, and J.-P. Seifert, “Having no mathematical model may not secure PUFs,” *J. Cryptograph. Eng.*, vol. 7, no. 2, pp. 113–128, 2017.

[47] F. Ganji, D. Forte, and J.-P. Seifert, “Rock‘n’roll PUFs: Crafting provably secure PUFs from less secure ones,” *J. Cryptographic Eng.*, vol. 11, pp. 33–48, May 2020.

[48] M. Barbareschi, A. D. Benedictis, and N. Mazzocca, “A PUF-based hardware mutual authentication protocol,” *J. Parallel Distrib. Comput.*, vol. 119, pp. 107–120, Sep. 2018.

[49] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, “Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response,” *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109–1123, Apr. 2019.

[50] P. Gope, J. Lee, and T. Q. S. Quek, “Lightweight and practical anonymous authentication protocol for RFID systems using physically unclonable functions,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2831–2843, Nov. 2018.

[51] M. A. Qureshi and A. Munir, “PUF-IPA: A PUF-based identity preserving protocol for Internet of Things authentication,” in *Proc. IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, 2020, pp. 1–7.

[52] F. Farha, H. Ning, K. Ali, L. Chen, and C. Nugent, “SRAM-PUF based entities authentication scheme for resource-constrained IoT devices,” *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5904–5913, Apr. 2021.

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

[54] P. Gope and B. Sikdar, “Lightweight and privacy-preserving two-factor authentication scheme for IoT devices,” *IEEE Internet Things J.*, vol. 6, no. 1, pp. 580–589, Feb. 2019.

[55] M.-D. Yu *et al.*, “A lockdown technique to prevent machine learning on PUFs for lightweight authentication,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Jul.–Sep. 2016.

[56] G. T. Becker, “The gap between promise and reality: On the insecurity of XOR arbiter PUFs,” in *Proc. CHES*, 2015, pp. 535–555.

[57] E. I. Vatajelu, G. Di Natale, M. S. Mispan, and B. Halak, “On the encryption of the challenge in physically unclonable functions,” in *Proc. IOLTS*, 2019, pp. 115–120.

[58] O. Günlü, O. Iscan, V. Sidorenko, and G. Kramer, “Code constructions for physical unclonable functions and biometric secrecy systems,” *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 2848–2858, Nov. 2019.

[59] B. Chen, T. Ignatenko, F. M. J. Willems, R. Maes, E. van der Sluis, and G. N. Selimis, “A robust SRAM-PUF key generation scheme based on polar codes,” in *Proc. Global Commun. Conf.*, 2017, pp. 1–6.

[60] O. Günlü, T. Kernetzky, O. Iscan, V. Sidorenko, G. Kramer, and R. F. Schaefer, “Secure and reliable key agreement with physical unclonable functions,” *Entropy*, vol. 20, no. 5, p. 340, 2018.

[61] E. Öztürk, G. Hammouri, and B. Sunar, “Towards robust low cost authentication for pervasive devices,” in *Proc. Pervasive Comput. Commn.*, 2008, pp. 170–178.

[62] S.-J. Wang, Y.-S. Chen, and K. S.-M. Li, “Adversarial attack against modeling attack on PUF,” in *Proc. DAC*, 2019, pp. 1–6.

[63] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. CCS*, 2002, pp. 148–160.

[64] Y. Shihong, L. Ping, and H. Peiyi, “SVM classification: Its contents and challenges,” *Appl. Math. A J. Chin. Univ.*, vol. 18, no. 3, pp. 332–342, 2003.

[65] K. Gurney, *An Introduction to Neural Networks*. Boca Raton, FL, USA: Taylor&Francis, 1997.

[66] A. A. Adewuyi, H. Cheng, Q. Shi, J. Cao, Á. MacDermott, and X. Wang, “CTRUST: A dynamic trust model for collaborative applications in the Internet of Things,” *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5432–5445, Jun. 2019.

[67] *Xilinx ARTIX-7 FPGA*. Accessed: Jul. 2020. [Online]. Available: https://digilentinc.com

[68] N. Hansen, “The CMA evolution strategy: A comparing review,” in *Towards a New Evolutionary Computation*. Heidelberg, Germany: Springer, 2006, pp. 75–102.

[69] *CMA-ES Attack*. Accessed: Jul. 2020. [Online]. Available: https://github.com/scluconn/DA_PUF_Library

[70] P. H. Nguyen *et al.*, “The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks,” in *Proc. CHES*, 2019, pp. 243–290.

[71] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, “Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs,” in *Proc. Int. Conf. Reconfig. Comput. FPGAs*, 2010, pp. 298–303.

[72] L. E. Bassham *et al.*, *A Statistical Test Suite for Random & Pseudorandom Number Generators for Cryptographic Applications*, NIST document SP 800-22, NIST, Gaithersburg, MD, USA, 2010.

[73] (2010). *Product Brief—JN5148 Module (Jennet, ZigBee Pro and IEEE 802.15.4 Module)*. [Online]. Available: https://www.glynstore.com/content/docs/jennic/JN5148-MO-PB_1v1.1.pdf

[74] J. M. Kim, H. S. Lee, J. Yi, and M. Park, “Power adaptive data encryption for energy-efficient and secure communication in solar-powered wireless sensor networks,” *J. Sensors*, vol. 2016, pp. 1–9, Mar. 2016.

[75] (2018). *On Performance*. [Online]. Available: https://www.bearssl.org/speed.html#measuring-speed

[76] B. Tsao, Y. Liu, and B. Dezfouli, “Analysis of the duration and energy consumption of AES algorithms on a Contiki-based IoT device,” in *Proc. 16th EAI Int. Conf. Mobile Ubiquitous Syst. Comput. Netw. Services*, 2019, pp. 483–491.

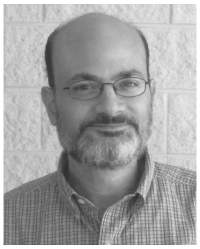
[77] L. Ertaul and A. Woodall, “IoT security: Performance evaluation of Grain, MICKEY, and Trivium—Lightweight stream ciphers,” in *Proc. IEEE Conf. Security Manag.*, 2017, pp. 32–38.

[78] C. Paar and J. Pelzl, *Understanding Cryptography—A Textbook for Students & Practitioners*. Heidelberg, Germany: Springer, 2010.

[79] S. Deb and B. Bhuyan, “Performance analysis of current lightweight stream ciphers for constrained environments,” *Indian Acad. Sci.*, vol. 45, p. 256, Oct. 2020.

**Mohammad Ebrahimabadi** (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering from Zanjan University, Zanjan, Iran, in 2008, and the M.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Maryland at Baltimore County, Baltimore, MD, USA.

He is a member of the Secure, Reliable and Trusted Systems Research Laboratory, University of Maryland at Baltimore County. His current research focus is on hardware security, and in particular side-channel analysis and fault injection attacks and countermeasures, sensor-assisted secure and reliable design, as well as developing PUF-based authentication and secure communication protocols in IoT frameworks.



**Mohamed Younis** (Senior Member, IEEE) received the Ph.D. degree in computer science from New Jersey Institute of Technology, Newark, NJ, USA, in 1997.

He is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC). Before joining UMBC, he was with Honeywell International Inc., Charlotte, NC, USA, where he led multiple projects for building integrated fault-tolerant avionics and dependable computing infrastructure. He also participated in the development of the Redundancy Management System, which is a key component of the Vehicle and Mission Computer for NASA's X-33 space launch vehicle. He has published about 300 technical papers in refereed conferences and journals. He has seven granted and three pending patents. His technical interest includes network architectures and protocols, wireless sensor networks, embedded systems, fault-tolerant computing, secure communication, and distributed real-time systems.

Prof. Younis serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences. He is a Senior Member of the IEEE Communications Society.

**Naghmeh Karimi** (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from the University of Tehran, Tehran, Iran, in 1997, 2002, and 2010, respectively.

She was a Visiting Researcher with Yale University, New Haven, CT, USA, from 2007 to 2009, and a Postdoctoral Researcher with Duke University, Durham, NC, USA, from 2011 to 2012. She has been a Visiting Assistant Professor with New York University, New York, NY, USA, and Rutgers University, New Brunswick, NJ, USA, from 2012 to 2016. She joined the University of Maryland at Baltimore County, Baltimore, MD, USA, as an Assistant Professor in 2017, where she leads the Secure, Reliable and Trusted Systems Research Laboratory. She has published three book chapters and authored/coauthored more than 60 papers in referred conference proceedings and journal manuscripts. Her current research interests include hardware security, VLSI testing, design-for-trust, design-for-testability, and design-for-reliability.

Dr. Karimi is a recipient of the National Science Foundation CAREER Award in 2020. She serves as an Associate Editor for *Journal of Electronic Testing: Theory and Applications* (Springer). She is also the Corresponding Guest Editor of the *Journal on Emerging and Selected Topics in Circuits and Systems*; special issue in Hardware Security in Emerging Technologies.