A Scalable Protocol Level Approach to Prevent Machine Learning Attacks on PUF-based Authentication Mechanisms for Internet-of-Medical-Things

Abstract—The Internet-of-Things (IoT) is becoming a revolutionary paradigm, moving towards ubiquity in day to day life and used in several applications such as smart healthcare systems, industry 4.0, critical infrastructure, etc. As with any concept that relies on wireless communication, authentication is of paramount importance when it comes to security considerations. Devices in many IoT applications are severely constrained in terms of computational resources and are thus unable to utilize many modern cryptographic methods for security purposes. Physically-Unclonable-Functions (PUFs) propose to solve this issue by allowing devices to generate unique and secure digital fingerprints at extremely low computational cost. However, currently PUFs are deemed insecure due to Machine-Learning-based modeling-attacks that can successfully clone the PUFs mathematically in order to impersonate them. To address all these requirements, this paper introduces a new lightweight and practical anonymous authentication protocol for IoT that is resilient against machine learning attacks on PUFs.

Index Terms—Mutual Authentication, IoT Device, PUF, Machine Learning Attacks.

I. INTRODUCTION

The Internet-of-Things (IoT) is revolutionizing almost every aspect of our life. For example, with the emergence of IoT environments, smart homes have been integrated into our daily life and people can remotely operate and control equipment in their homes (e.g., people can adjust their home temperature when they are outside). The ubiquitous interconnection of physical objects greatly accelerates collection, aggregation and sharing of data with other connected devices and applications [1], [2], [3], thus widening IoT applicability in different domains ranging from smart healthcare, smart homes, industrial automation, etc. As a key application area of IoT, the Internet-of-Medical-Things (IoMT) [4] forms a new paradigm for healthcare through connected sensors, wearable medical devices, and clinical systems that are capable of improving the quality of medical care with reduced cost and more timely responses. The vision of “anywhere, anytime” access to healthcare is changing our expectations of healthcare and fueling future telehealth systems where health can be monitored remotely via health apps, diagnosis can be delivered via video consultations with clinicians, doctors can remotely diagnose and prescribe medicine, offer medical interventions, provide check-ups/monitoring and even carry out surgery. Transforming community health and care to use real-time information to support self-management of health and well-being, and to facilitate timely interventions is one of the four grand challenges of healthcare technologies. Figure 1 shows an IoMT-based system architecture, which consists of the following major entities: a set of medical IoT devices (attached with the patient), IoT gateway, and IoMT service provider. The IoMT devices collect data from the patient and send them to the IoMT service provider, which comprises of three components, i.e., an authentication server, a decision support system (DSS), and an information database (IDB). The server is responsible for validating the legitimacy of the IoMT devices along with the physical security of the devices worn by the patient. Next, upon receiving the data from the IoMT devices (via IoT gateway), the DSS will analyze the data and take appropriate actions. The IoMT service provider also consists of an information database, where it maintains the historical data of the patients for further analyses.

A. Importance of Security in IoMT

Security is one of the most important aspects for any communication environment. Usually, security is a collection of deployed mechanisms to protect a system as a whole. Most of the communications in IoMT based healthcare system are wireless in nature. This may cause severe threats to the security of the system. For example, these threats and attacks can create serious problems in the social life of an individual who uses this system, as this may reveal their confidential healthcare information to unauthorized parties. People with malicious intent may use confidential healthcare data to cause
harm and breach an individual’s privacy, for example, by tracking the location of a patient or publicly disclosing medical records. Various attacks such as “replay”, “man-in-the-middle”, “impersonation”, “illegal session key computation”, “password guessing” and other forms of data leakage attacks [5] are possible in such a communication environment. Therefore, it is essential to develop protocols (e.g., access control and key management) to secure the communication in an IoT based healthcare system. Apart from communication security, ensuring the physical security of the IoT device is also highly imperative because if these devices are physically tampered, then they will provide incorrect readings during patient monitoring, potentially causing a fatality. In order to deal with this important issue, the concept of physical unclonable functions (PUFs) [3] has been suggested as a security primitive in existing literature. However, based on the current research, PUFs have been shown to be vulnerable to modeling attacks [6], [7], [8], [9].

B. Related work and Motivation

Usually, IoT deployments consist of a collection of low cost and resource-constrained devices operating in unsupervised environments. 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. Some interesting PUF-based lightweight authentication protocols for IoT devices have been proposed in the literature, such as [10-17]. Although these authentication schemes are lightweight and benefit from the unique footprints of PUF devices, they suffer from vulnerabilities such as modeling attacks [6-9]. Ganji et al. [34]-[36] 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) bit(s). 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.

To safeguard PUF-based authentication solutions against possible modeling attacks, Yu et al. [20] introduced two authentication protocols. In order to secure their PUF against modeling attacks, their proposed solutions limit the number of CRPs that can be transferred in the IoT framework. Besides, to restrict the possibility of reliability attacks where an attacker may try to exploit measurement noise to model the PUF, the protocols presented in [20] do not allow any repetitive challenge bit-streams. However, the proposed protocols are only suitable 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. In 2019, Liang et al. [22] presented another machine-learning resilience authentication protocol using train-model [21]. However, the protocol cannot ensure security against several imperative attacks such as replay attacks and impersonation attacks. Besides, similar to [20], the protocol presented in [22] suffers from the scalability issue.

Our goal and contribution: In this paper, we seek to address all the above issues related to security against machine learning attacks and scalability. In order to do that, we propose a new anonymous authentication scheme for IoT using PUFs that is secure against machine learning or modeling attacks. In this regard, we utilize the concept of one-time-PUF (OPUF), where after execution of each session of the protocol, the behavior of the PUF changes by using a reconfiguration process. The proposed scheme can not only ensure several imperative security properties (such as privacy of the devices and resistance to DoS attacks), but also provides higher degree of scalability and practicality. In a nutshell, the major contributions of this paper can be summarized as follows:

- We propose a new privacy preserving mutual authentication protocol for IoT devices which can prevent machine learning or modeling attacks on PUFs. Apart from that, the proposed scheme can also ensure higher degree of scalability, along with security against man-in-the-middle attacks, privacy of the IoT devices, and protection against forgery and replay attacks. To the best of our knowledge, this is the first PUF-based authentication protocol which can ensure security against machine learning or modeling attacks, along with privacy of the IoT devices.
- We provide results of high-level OPUF simulations to show the backward and forward unpredictability behavior of the PUF used in the protocol that provides resilience against ML or modeling attacks.
- We provide a rigorous security analysis of our proposed scheme to show that it is secure against some of the imperative attacks.

The rest of the paper is organized as follows. In Section II, we provide a brief introduction to machine learning attacks on PUFs, OPUF-security, and fractional Hamming distance. In Section III, we present the proposed privacy-preserving machine-learning resilience authentication scheme. Security of the proposed scheme is analyzed in Section IV. A relevant discussion based on the proposed scheme is provided in Section V with concluding remark in Section VI. All the symbols and cryptographic functions of the proposed scheme are presented in Table I.

II. Preliminaries

A. Machine Learning Attacks on PUFs

Physically Unclonable Functions are a promising security primitive that can be utilized within lightweight authentication protocols to facilitate high levels of security while simultaneously minimizing computational resource requirement per device. A PUF can be considered as a physical feature of a device; alternatively, it is equivalent to bio-metric features of humans, such as retina, fingerprints, etc. The built-in features of a particular PUF can never be reproduced or cloned due to its physical signature or makeup. In mathematical terms, it is regarded as a function that takes an input (Challenge) in the form of a bit string and returns unique output (Response) [4-5].

The PUF-based function can be expressed as $R ← PUF(C)$, where the variables $C$ and $R$ serve as the challenge-response pair (CRP). A PUF always returns the
same R for the same challenge, if tested again and again. Machine learning has been known to be a powerful threat to PUF by facilitating modeling attacks. These types of attacks generally involve an adversary collecting a large subset of a PUF’s possible CRPs: \( \{ (c_1, r_1), (c_2, r_2), \ldots, (c_w, r_w) \} \). From this, a mathematical model, \( \hat{m} \), can be derived as an algorithm in order to predict an unknown response, \( r_{w+1} \), to a new challenge, \( c_{w+1} \) [9]. As a result, most often only strong PUFs are susceptible to modeling attacks, with exceptions for weak PUFs where part of their obfuscation relies on interaction with a strong PUF[6][25].

B. One-Time PUF (OPUF)

Loosely speaking, a “One-Time PUF (OPUF)” is a strong PUF with a mechanism to change the PUF configuration or settings after each session of the authentication process. The idea of OPUFs stems from the literature surrounding resetting a PUF’s configuration [23-24]. Resetting describes a feature of a PUF that enables a complete change of individual behavior in response to challenges, by updating its state. OPUFs can achieve both forward- and backward-unpredictability: The former assures that responses measured before the reconfiguration event are invalid thereafter, while the latter assures that an adversary with access to a reconfigured PUF cannot estimate the PUF behavior before reconfiguration. Assuming that an adversary needs to collect a large subset of all of a strong PUF’s possible CRPs in order to mount a successful modeling attack, the resetting of an individual PUF would render such attacks useless. An adversary would then need to collect a new subset of CRPs for the new configuration in order to generate a new mathematical model for the attack. Additionally, it is assumed that the outcome of the resetting mechanism is uncontrollable and difficult to revert, even with invasive means. Also, after resetting, the configuration of the PUF does not affect the security properties of the original PUF (such as tamper detection, unclonability, etc.).

1) Example of OPUF: A DPUF can be considered as one of the ways to create an OPUF. In this section, we provide a brief description of DPUF. DRAM based PUFs such as DPUF are strong PUFs suitable for authentication systems as they have very large address spaces and high density to fulfill the requirement for enabling large numbers of CRPs for use during authentication. A DPUF is a PUF implementation that utilizes power-up and cell refresh behavior to generate entropy for transforming stored challenge bit-strings to response bit-strings, alongside a reconfiguration characteristic. DRAM modules store data in ‘bit-cells’ consisting of capacitors and access transistors, which hold either a value of ‘0’ or ‘1’, depending on the charge state of the respective capacitor. These capacitors, however, leak electrical charge over a period of time and must be periodically refreshed in order to maintain data integrity. The required interval between refreshes remains a standard of 62ms, which is referred to as the refresh-pause interval. A DPUF exploits this mechanism by increasing the refresh pause interval to cause deliberate errors in the form of bit-flips in some of the cells, randomly. As a result, a bit-string challenge may be stored within an arbitrary block in a DRAM module and then the refresh-pause interval applied, with the result being a new bit-string stored across the memory block that can be read as a ‘final response’. Adjusting parameters such as the refresh-pause timing or varying the allocated memory block allows for new challenge-response behavior, meaning that the block ‘state’ can effectively be reconfigured. Sutar et. al. [23] also tested the robustness and uniqueness property of DPUF reconfiguration under varying operating conditions such as temperature and aging. Assuming a minimum entropy of 400 bit-flips across 512 blocks in the module, in both regular and irregular conditions of the DPUF, refresh-pause intervals of 40-60 seconds across each block generate entropy in excess of 1200 bit flips. As the entropy variation across these intervals is high, refresh-pause intervals are validated as an appropriate mechanism for ensuring entropy. This resulting bit-flip behavior is extremely difficult to predict, meaning any modeling attempt on a previous attempt has no advantage in predicting the CRPs for any new state.

2) OPUF Security against Machine Learning: For many PUF implementations, especially those whose functionality is attributed to a linear delay model such as the arbiter PUF [28], security is most often impacted by adversaries attempting modeling attacks that utilize machine learning techniques [6], [9]. These attacks attempt to (with high levels of success) generate a highly accurate model clone of a physical PUF in order to predict the challenge-response behavior, and thus output correct unknown responses to new challenges. To be successful, these attacks require a large number of known CRPs to train the model. With an OPUF (e.g., DPUF), however, this type of attack is extremely difficult to carry

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( OID_T^i )</td>
<td>One-time identity of device ( T ) for ( i )-th round</td>
</tr>
<tr>
<td>( CRP(c_i, R_i) )</td>
<td>Challenge-response pair for the ( i )-th round</td>
</tr>
<tr>
<td>( R_i )</td>
<td>Round-key for the ( i )-th round</td>
</tr>
<tr>
<td>( WPUF_T )</td>
<td>Weak PUF attached with the ROM-BIOS of device ( T )</td>
</tr>
<tr>
<td>( OPUF_T^i )</td>
<td>One-time PUF Configuration for ( i )-th round of device ( T )</td>
</tr>
<tr>
<td>( h(\cdot) )</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>( \oplus )</td>
<td>Exclusive-OR operation</td>
</tr>
<tr>
<td>( | )</td>
<td>Concatenation operation</td>
</tr>
</tbody>
</table>

### Table I: SYMBOLS AND CRYPTOGRAPHIC FUNCTIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( OID_T^i )</td>
<td>One-time identity of device ( T ) for ( i )-th round</td>
</tr>
<tr>
<td>( CRP(c_i, R_i) )</td>
<td>Challenge-response pair for the ( i )-th round</td>
</tr>
<tr>
<td>( R_i )</td>
<td>Round-key for the ( i )-th round</td>
</tr>
<tr>
<td>( WPUF_T )</td>
<td>Weak PUF attached with the ROM-BIOS of device ( T )</td>
</tr>
<tr>
<td>( OPUF_T^i )</td>
<td>One-time PUF Configuration for ( i )-th round of device ( T )</td>
</tr>
<tr>
<td>( h(\cdot) )</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>( \oplus )</td>
<td>Exclusive-OR operation</td>
</tr>
<tr>
<td>( | )</td>
<td>Concatenation operation</td>
</tr>
</tbody>
</table>
out and has a very low probability of success for multiple reasons. First, if the authentication system utilizes a different DRAM block for each round of authentication, an attacker cannot gain insights into the challenge-response behavior of a new block based on the knowledge of challenge-response pairs of a previous block. This is due to the unique, random bit-flip entropy across different blocks in a DRAM. In addition, DPUF reconfiguration ensures further security in this regard. For example, assume a scenario where an adversary is able to collect complete information about the bit-flip behavior of a given block (128KB size) for a 60s interval. In the next round of authentication, the same block is used, however, this time with a 40s refresh pause interval. With this same block, 1200-1400 new bit flips would be generated, rendering any modeling for the previous block useless. The probability of correct response prediction would thus be extremely small, noting that an attacker can only attempt authentication requests a limited number of times.

C. Fractional Hamming Distance (FHD)

Because of their physical nature, PUFs are inherently noisy. As a result, for a given challenge \( C \), the PUF output \( R \) will differ slightly when it is measured multiple times. This gap can be expressed by the Hamming distance, which is a popular metric used in error correction of noisy outputs from PUFs. For a given string of fixed length, the Hamming distance can determine the difference between it and another string of the same length by measuring the number of substitutions required to change the given string into the other. Two identical strings would thus have a Hamming distance of zero. Now, considering binary vectors \( r \) and \( \tilde{r} \) of same length. Then, the Hamming weight \( \text{HW}(r) \) counts the number of non-zero digits (i.e., 1’s) in the vector \( r \), and the fractional Hamming distance is given by \( \text{FHD}(r, \tilde{r}) = \frac{\text{HW}(r \oplus \tilde{r})}{L(r)} \), where \( L(r) \) denotes the length of \( r \).

III. PROPOSED ML-ATTACK PREVENTION-BASED AUTHENTICATION SCHEME

The proposed scheme consists of two phases: registration or setup phase and the authentication phase. In the registration phase, each IoT device \( T \) is required to register with the server. During authentication, each device will prove its legitimacy to the server and similarly, the server will also prove its trustworthiness to the device.

A. Adversary Model

In order to show the potential security of an authentication scheme, it is important to define the adversary’s capabilities through an adversary model. For our proposed PUF-based authentication scheme, we consider the following capabilities of the adversary. First, in our proposed scheme, we allow an adversary to eavesdrop on the communication channel between the server and the device. He/she may also change and block some of the messages sent between the two entities. Manipulation and replay of protocol traffic are also deemed to be possible. Additionally, we also allow the adversary to mount physical and cloning attacks on the PUF. Finally, in the proposed scheme we consider the added ability of the adversary to attempt machine learning modeling attacks (as described in Section II.D).

B. Assumptions

In our proposed scheme, we assume that the operations of the registration phase are carried out over a secure channel. After that, the adversary has access to the device’s interface and is free to execute brute-force queries, with the possibility that the challenges are chosen adaptively. The obtained CRP information can then be used for algorithmic machine learning attacks. Now, the proposed scheme uses two PUFs: (i) a weak PUF (WPUF) (such as SRAM PUF) that uses the memory cells of the SRAM that is part of the System-on-Chip (SoC) that includes the CPU of the IoT device, and (ii) a OPUF (which is basically a strong PUF with reconfigurable property such as DPUF [23]) attached with the device’s external (i.e, off
chip) memory. Thus, we assume that the WPUF is physically inaccessible to the adversary but the adversary has physical access to the OPUF to obtain a considerable number of CRPs. However, any attempt to tamper with a PUF will change the behavior of the PUF and render the PUF useless. The server has secure storage available and its computation are protected from the outside world. Only one OPUF instance can be used per protocol session. The OPUF instance is bound to a particular session and cannot be used in any other session.

C. Registration Phase

The registration phase consists of the following steps:

**Step Reg₀:** The server randomly generates two challenges \( \{ C_i, C_2 \} \) and sends them to the device \( T \) through a secure channel.

**Step Reg₁:** After receiving the challenges \( \{ C_i, C_2 \} \), device \( T \) generates a unique random key \( K_{i-1} \) and then the device uses its weak PUF (WPUF) and computes \( R_x = \text{WPUF}_T(C_x) \), along with the key for the \( i \)-th round of authentication, i.e., \( K_i = h(R_x||K_{i-1}) \). After that, the device uses its strong one-time PUF (OPUF) and extracts the PUF-output \( R_i = \text{OPUF}_T(C_i) \), where \( R_i \) can be divided into two-parts, i.e., \( \{ R_1, R_2 \} \). Hereafter, the device composes a message with the following parameters \( \{ ID_T, R_i, K_i, R_x \} \) and sends it to the server through the secure channel, where \( ID_T \) denotes the real identity of the device.

**Step Reg₂:** Next, the server uses its master key (MK) and computes the one-time-identity \( OID_T = h(R_i||ID_T||MK) \) for the \( i \)-th round of authentication and sends \( OID_T \) to device \( T \) through the secure channel. Then, the server stores \( \{ OID_T, \{ C_i, R_i, K_i, R_x \} \} \) for authenticating the device in the \( i \)-th round.

**Step:** After receiving the one-time-identity \( OID_T \), the device stores \( \{ OID_T \} \) in its memory and also stores \( \{ K_{i-1}, C_2 \} \) in the ROM-BIOS of the device which is attached with the WPUF. Details of the registration phase of the proposed scheme are depicted in Fig. 1.

D. Authentication Phase

The \( i \)-th round of the authentication phase of the proposed scheme consists of the following steps:

**Step 1:** Device \( T \) first selects its one-time-identity \( OID_T \) and also generates a random number \( N_t \). After that, the device loads \( \{ K_{i-1}, C_2 \} \) into its memory from the ROM-BIOS and computes \( R_x = \text{WPUF}_T(C_x) \), \( K_i = h(R_x||K_{i-1}) \), \( N_t = N_t \oplus K_i \), and \( V_0 = h(N_t||K_i) \). Finally, the device composes a message \( M_1 : \{ OID_T, N_t, V_0 \} \) and sends it to the server for verification.

**Step 2:** Upon receiving the authentication request message \( M_1 \) from the device, the server first finds \( OID_T \) in its database. If the server cannot find \( OID_T \) in its database, then it aborts the authentication process. Otherwise, the server reads \( \{ C_i, R_i, K_i, R_x \} \) and computes and verifies the hash response \( V_0 \). If the verification is successful, then the server generates a random number \( N_s \) and computes \( N_s = N_s \oplus K_i \), \( N_s = K_i \oplus N_s \), \( R_1 = R_1 \oplus K_i \), and \( V_1 = h(N_s||K_i||R_1||N_t) \). Finally, the server composes a message \( M_2 : \{ C_i, R_1, V_1, N_s \} \) and sends it to the device.

**Step 3:** Upon receiving message \( M_2 \), the device first computes \( N_s = K_i \oplus N_s \) and \( R_1 = R_1 \oplus K_i \), and verifies the parameter \( V_1 \). If the verification is not successful, then the device aborts the execution of the protocol. Otherwise, the device generates \( \{ R_1||R_2 \} = \text{OPUF}_T(C_i) \)
and also checks whether \( \text{FHD}(R_{i+1}^2, R_i^2) > \tau \). If so, the device terminates the execution of the protocol. Otherwise, the device computes \( X = R_i^2 \oplus K_i, C_{i+1} = h(C_i || N_i || K_i) \) and resets (reconfigures) the strong-PUF for the \((i+1)\)-th round of authentication. Subsequently, the device extracts the PUF output \( R_{i+1} = \text{OPUF}_{T+1}^2(C_{i+1}) \), and then computes \( R_{i+1} = R_{i+1} \oplus K_i, OID_{T+1}^2 = h(OID_{T+1} || R_{i+1}) \) for the \((i+1)\)-th round of authentication. Hereafter, the device computes \( V_2 = h(K_i || R_{i+1} || N_i || X) \) and composes a message \( M_3 = \{ R_{i+1}^2, X, V_2 \} \) and sends it to the server. Finally, the device stores \( OID_{T+1}^2 \) in its memory and replaces \( K_{i-1} \) with \( K_i \) stored in the ROM-BIOS attached with the WPUF of the device.

**Step 4:** Next, when the server receives message \( M_3 \) from the device, then the server first computes and verifies the parameter \( V_2 \) in order to check the integrity of the other parameters in \( M_3 \) and also to validate the legitimacy of the device. If the validation is successful, then the server computes \( R_i^2 = X \oplus K_i \), and checks whether \( \text{FHD}(R_i^2, R_i^2) > \tau \).

If not, then the server computes \( C_{i+1} = h(C_i || N_i || K_i), R_{i+1} = R_{i+1} \oplus K_i, OID_{T+1}^2 = h(OID_{T+1} || R_{i+1}) \), and \( K_{i+1} = h(K_i || R_e) \) for the \((i+1)\)-th round of authentication. Finally, the server replaces \( K_i \) with \( K_{i+1} \), \( C_i \) with \( C_{i+1} \), \( R_i \) with \( R_{i+1} \), and \( OID_{T+1}^2 \). Therefore, the server stores \( \{ OID_{T+1}^2, \{ C_{i+1}, R_{i+1}, K_{i+1} \} \} \) for authenticating the device in the \((i+1)\)-th round. Details of the authentication phase of the proposed scheme are depicted in Fig. 2. Now, to handle any desynchronization between the device and the server, during the registration phase, apart from \( \{ C_i, C_e \} \), the server needs to generate a set of synchronization (SYN) challenges \( C_{SYN} = \{ C_{SYN}^1, \ldots, C_{SYN}^n \} \) and send them to the device. After that, in Step Reg, the device uses \( \text{OPUF}_T \) and generates \( R_{SYN} = \{ R_{SYN}^1, \ldots, R_{SYN}^n \} \leftarrow \text{OPUF}_T(C_{SYN} = \{ C_{SYN}^1, \ldots, C_{SYN}^n \}) \) and also generates a few Reference ID and SYN Key pairs, i.e., \( (Ref_{ID, K_{SYN}}) = \{ (Ref_{ID}^1, K_{SYN}^1), \ldots, (Ref_{ID}^n, K_{SYN}^n) \} \) and subsequently sends \( (Ref_{ID}, R_{SYN}, C_{SYN}, R_{SYN}) = \{ (Ref_{ID}^1, K_{SYN}^1, C_{SYN}^1, R_{SYN}^1), \ldots, (Ref_{ID}^n, K_{SYN}^n, C_{SYN}^n, R_{SYN}^n) \} \) to the server through the secure channel and also keeps a copy in its ROM-BIOS. Note that for restricting any modeling attack, the device needs to use a new set of the PUF parameters for generating each \( C_{SYN}, R_{SYN} \). Now, in case of loss of synchronization, both the server and the device need to use one of the sets \( (Ref_{ID}^1, K_{SYN}^1, C_{SYN}^1, R_{SYN}^1) \) from \( (Ref_{ID}, K_{SYN}, C_{SYN}, R_{SYN}) \). Once a set is used up, it must be deleted from both the ends. In this regard, during the execution of the synchronization phase, both the device and server need to use \( K_{SYN}^i \) and \( Ref_{ID}^i \) in the same way as \( K_i \) and \( OID_T^i \) are used in the \( i \)-th session. In this way, we can address any desynchronization between a device and the server without compromising the privacy. Finally, an IoT-based application may require both the device and the server to generate a session key, so that they can communicate securely. In this regard, once both the device and the server authenticate each other through the execution of the above steps of the proposed scheme, they can compute the session key for the \( i \)-th session as \( SK_i = h(N_i || N_s || R_i^1 || R_s^1) \).

---

**IV. Security Analysis**

In this section, we provide a discussion to show that the proposed authentication protocol can ensure some of the imperative security properties such as mutual authentication, privacy of the devices, etc.

1) **Mutual Authentication:** In the proposed authentication scheme, a device \( T \) can compute \( N_i^* = N_i \oplus N_i \) and also obtain \( N_s = K_i \oplus N_i^* \) and \( R_i^2 = R_i^2 \oplus K_i \), when it is legitimate. Here, the server authenticates the device by checking the key-hash response \( V_0 = h(N_i^* || K_i) \) and \( V_2 = h(K_i || R_{i+1} || N_i || X) \) and if the fractional Hamming distance \( \text{FHD}(R_i^2, R_i^2) < \tau \). Conversely, the device can authenticate the server by using the key-hash response parameter \( V_1 \) and whether the fractional Hamming distance \( \text{FHD}(R_i^2, R_i^2) < \tau \), where a legitimate server only knows the secret credentials \( R_i \) and \( K_i \). In this way, the proposed scheme can satisfy the mutual authentication property.

2) **Privacy of the Devices:** In the proposed authentication scheme, for any two consecutive sessions \( i \) and \( i+1 \), a device needs to use two different one-time-identities, \( OID_T^i \) and \( OID_T^{i+1} \) (i.e., \( OID_T^i \neq OID_T^{i+1} \)). No device is allowed to use the same one-time-identity twice. Moreover, all the parameters in \( M_1, M_2 \) and \( M_3 \) are for one-time use. Thus, if the adversary \( A \) intercepts these transmitted messages for two consecutive sessions \( i \) and \( i+1 \), even then he/she will not be able to differentiate them from randomly chosen strings.

On the other hand, in case of loss of synchronization, the device needs to use one of the unused Reference ID and SYN Key pairs, i.e., \( (Ref_{ID}, K_{SYN}) = \{ (Ref_{ID}^1, K_{SYN}^1), \ldots, (Ref_{ID}^n, K_{SYN}^n) \} \). Subsequently, both the device and the server need to delete them from their memory. Hence, we can state that changing the identities in each session and the randomness of the transmitted parameters parameters in \( M_1, M_2 \) and \( M_3 \) ensures privacy against eavesdropping (PAE) attacks.

3) **Forward Secrecy:** The secret credentials such as \( R_i \) and \( K_i \) are valid only for a specific session. After that, both of them are updated with \( R_{i+1} \) and \( K_{i+1} \), where \( R_{i+1} \) is generated by resetting the OPUF’s configuration and \( K_{i+1} = h(K_i || R_e) \). Now, even if the adversary \( A \) knows \( R_i \) and \( K_i \), it will be difficult for him/her to compute \( R_{i+1} \) and \( K_{i+1} \). In order to do that \( A \) needs to know the internal state of the OPUF, and the secret \( K_{i-1} \).

4) **Resilience Against Modeling or Machine-Learning Attacks:** The proposed scheme is based on the dynamic OPUF, where after each round the authentication process, the PUF’s configuration is updated. Now, if an adversary \( A \) has access to the device and is provided with a set of CRPs, it may develop a model for the OPUF. However, the OPUF’s behavior is changed after each resetting operation. Hence, it will be difficult for \( A \) to perform any modeling or ML attacks.

5) **Protection Against Physical Attacks:** Assume that an
attacker wants to do physical tampering on an IoT device for his/her own profit. However, since any such an attempt directly reflects on the PUF’s behavior, the verifier will be able to detect it. Now, in the proposed scheme, if \( A \) tries to access the secret credentials such as \( K_{i-1} \) and \( C_x \) stored in the ROM-BIOS attached with the WPUF, then the behavior of the WPUF will be changed. As a result, the WPUF will not be able to generate the desired output \( R_x = \text{WPUF}_T(C_x) \), and thus the device will not be able to generate the desired \( K_i = h(R_x||K_{i-1}) \) and \( V = h(N_i^*||K_i) \). Therefore, the server will be able to detect that. On the other hand, if the adversary \( A \) tries to access the device memory attached with the OPUF, then the behavior of the OPUF will be changed. The server will be able to comprehend such an attempt of tampering by using the parameter \( V_2 = h(K_i||R_{i+1}^*||N_x||X) \) and fractional Hamming distance \( \text{FHD}(R_i^2, R_i^2) \). In this way, the proposed scheme can ensure security against any physical attacks on an IoT device.

6) Protection Against Replay Attacks: In the proposed authentication scheme, if an adversary \( A \) tries to resend the message \( M_1 : \{OID_T, N_i^*, V_0\} \), then he/she will not be successful since \( OID_T \) and \( V_0 \) change in each session (due to updated \( K_i \)). Next, if \( A \) tries to resend message \( M_2 \), then he/she will not be successful, since a new challenge \( C_i \) is used in each session and the key-hash response \( V_1 \) is generated with the nonce \( N_i \), which was sent to the server in encoded way (where only a legitimate server can decode \( N_i = N_i^* \oplus K_i \)). Finally, \( A \) cannot reuse message \( M_3 \), since it consists of a new response where the parameter \( X \) contains \( R_{i+1}^* \), which is valid only for the \( i \)-th session. Moreover, \( V_2 \) contains the nonce \( N_e \), which was sent in \( M_2 \) after being encoded. In this way, the proposed authentication scheme can ensure security against replay attacks.

V. DISCUSSION

A. Performance Comparison

In this section we compare our proposed machine-learning resilience authentication protocol with other related schemes, such as the schemes of Yu et al. [20] and Liang et al. [22]. In order to analyze the performance of the proposed scheme, particularly on the security front, our scheme has been compared with two protocols presented in [20] and the protocol presented in [22], by considering the major security properties (as shown in Table II). From Table II, we can see that the protocols presented in [20] and [22] cannot ensure most of the imperative security properties such as protection against replay attacks, protection against tracking attacks, etc. Besides, the Protocol:1 presented in [20] and the proposed scheme can ensure security against any learner. In contrast, Protocol:2 presented in [20] and [22] ensure security against heuristic learners. Nevertheless, only the proposed protocol can ensure scalability. Conversely, the two protocols presented in [20] and the protocol presented in [22] can only support a limited number of sessions.

### B. OPUF Backward and Forward Unpredictability

Many proposed PUF-based authentication protocols rely on non-reconfigurable PUF implementations [20],[22]. As a result, for example, if a successful modeling attack on a specific device’s PUF is mounted, future challenges may be predicted and this does not ensure forward unpredictability. In addition, in the case that a PUF is used for key generation, an old used key can potentially be collected to decrypt previously sent encrypted traffic on a network, thus not ensuring backward unpredictability. By designing our protocol around state reconfigurable behavior, we can guarantee these features. Consider a non-reconfigurable PUF construction such as the arbiter PUF that has an unchangeable behavior which we can refer to as the state \( S_i \). If an adversary can model state \( S_i \), then she/he may predict \( R_i \) for a given \( C_i \), i.e., functionally: \( R_i \leftarrow \text{PUF}_S(C_i) \). In the case of OPUF, which is inspired by the refresh function of the DPUF [23] (a description of DPUP is given in the Appendix), we can consider change across these states through reconfiguration. Consider three rounds of authentication where the OPUF has issued responses to challenges based upon three given states: \( S_{i-1}, S_i, \) and \( S_{i+1} \). A given adversary that has access to a correct model of \( \text{PUF}_{S_i} \), she is able to determine \( R_i \) for any given \( C_i \). However, now she wishes to determine \( R_{i-1} \) for a previous OPUF state \( S_{i-1} \), or, \( R_{i+1} \) for a forthcoming state \( S_{i+1} \). In this scenario, her model can only accurately predict CRPs for state \( S_i \) and thus has no extra advantage when attempting a prediction for these other states as her model is not trained with adequate CRPs for these states. In this case, the adversary must attempt to model every given state for every round of authentication, which is highly infeasible and impractical.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol #1 of [20]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Protocol #2 of [20]</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Protocol of [22]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Proposed Scheme</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

SF1: Man-in-the-Middle-Protection; SF2: Replay-Attacks-Protection; SF3: DoS-Attack-Protection; SF4: Security-Against-Any-Learner; SF5: Tracking-Attack-Protection; SF6: Scalability; ✓: Yes; ×: No
Algorithm 1: High Level OPUF Simulation

Initialization:
\[
OPUF^c \leftarrow \begin{pmatrix}
0,1 & \ldots & 0,1 \\
\vdots & \ddots & \vdots \\
0,1 & \ldots & 0,1
\end{pmatrix}
\]
\[
OPUF^r \leftarrow OPUF^c
\]
\[
R \leftarrow \{True, False\}
\]
if \( R == true \) then
\[
S \leftarrow PRNG(1);
\]
for \( f + 1 \) do
\[
x \leftarrow PRNG(S, b + 1)
\]
\[
y \leftarrow PRNG(S, b + 1)
\]
if \( OPUF^r[x][y] == 1 \) then
\[
OPUF^r[x][y] \leftarrow 0
\]
else
\[
OPUF^r[x][y] \leftarrow 1
\]
Output:
\[
OPUF^r
\]

C. OPUF Simulation Results to Demonstrate Backward and Forward Unpredictability

To demonstrate the backward and forward unpredictability of our protocol, we present simulation results of OPUF behavior. We consider an OPUF based on DPUF. The simulation code was written in Python and executed on an Intel i5-6600 quad-core 3.30GHz CPU system with 16GB DDR4 system memory. We simulated a challenge/response mechanism based on a DPUF with two-dimensional arrays to emulate the matrix structure of DRAM cell blocks. In addition, we utilize pseudo-random number generation (PRNG) for simulating the bit-flips that occur during the refresh/pause interval of a DPUF. Algorithm 1 describes the functioning of our OPUF simulation. An initial two-dimensional array \( OPUF^c \) of block size \( b \times b \) is chosen where each element stores a value \( \in \{0, 1\} \). A seed \( S \) is used for the PRNG function to model the effect of state reconfiguration. If the OPUF is to have its state reconfigured \((R = True)\), a new seed \( S \) is generated. For \( PRNG(S) \) number of elements \( f \) within the block size entropy threshold, the stored bit is flipped \( 1 \leftarrow 0 \) or \( 0 \leftarrow 1 \). The block size entropy threshold is determined by the block size chosen: \( 128KB = 1400-1800 \) bits, \( 64KB = 800-1000 \) bits and \( 32KB = 300-500 \) bits. The resulting two-dimensional array \( OPUF^r \) forms the final response and is appended with the challenge \( OPUF^c \) to form a CRP. We illustrate the nature of the reconfiguration property of the OPUF using three seeds corresponding to three reconfigurations. In the first round of authentication, given a seed ‘123’ \((S_{i-1})\) and a challenge, in Figure 4 we observe a corresponding response that is intrinsic to state \( S_{i-1} \). This observation on its own is functionally identical to that of any other type of non-reconfigurable PUF (such as XOR APUF) in terms of \( R \leftarrow PUF(C) \). Now, consider a reconfiguration of the OPUF by using a different seed ‘567’, corresponding to state \( S_i \). For the same challenge that was used previously, the OPUF now provides a new response (i.e., the new state has a new bit/flip entropy): \( R_{i+1} \leftarrow PUF_{i+1}(C) \). Similarly, another reconfiguration of the OPUF by using the seed ‘789’ produces yet another new response for the same challenge, based on new bit-flip entropy for what is state \( S_{i+1} \) for the third round of authentication. Figure 5 demonstrates the bit-flip entropy of the OPUF through Hamming distance values, where the X-axis denotes the index of each OPUF reconfiguration instance and the Y-axis denotes the Hamming distance between each CRP. We generated 150 individual OPUF reconfigurations for each designated block size of 32KB, 64KB and 128KB (450 in total), using randomly chosen seeds between 0 and 200000. As can be seen, the larger block sizes generate higher levels of bit-flip entropy, resulting from the underlying operating principles of DPUFs. A 32KB block size produces a minimum and maximum Hamming distance between 200 and 500, 64KB block produces between 800 and 1000, and 128KB block produces between 1400 and 1800.

D. Formal Security Verification Using ProVerif Simulations

In order to verify the security and robustness of the proposed security protocol in terms of the specific goals such as mutual authentication, secure-key-exchange etc., we performed a formal proof using ProVerif [26]. ProVerif is an efficient
and stable analytical tool that can infer concepts based on specific rules, and is widely used in the analysis of information that involves the confidentiality and security of protocols. The weak PUF and one-time PUF are defined as the functions $WPUF$ and $OPUF$, respectively. In addition, they are defined as functions that accept a bit-string and return a bit-string, that are also inaccessible to an adversary, i.e., their behavior cannot be predicted. The bitwise XOR function is defined as $xor$, and it accepts two bit-strings and returns a single bit-string. The pre-definition part of the algorithm is shown in Fig. 6. The results (Fig. 7) show that the proposed protocol successfully executed two-way authentication in ProVerif, indicating that the protocol is secure in ProVerif environments.

VI. CONCLUSION

PUFs have gained popularity in the security domain as an useful paradigm for hardware-based root of trust for resource constrained IoT devices. At the same time, machine learning or modeling attacks have also become one of the main security challenges for PUF-based security solutions. Although a few protocol level approaches have been proposed in literature, they are not scalable, making them unsuitable for IoT environments. Therefore, in this article, we propose a scalable protocol level approach which can ensure security against any modeling or machine-learning attacks. Through comprehensive security analyses, we showed that apart from resilience against any machine-learning and modeling attacks, the proposed scheme can also ensure several imperative security features such as security against man-in-the-middle attacks, forward secrecy, privacy of the IoT devices, etc. Hence, it can be argued that the proposed scheme is a viable and promising solution for the security of IoT devices.
REFERENCES


