Lightweight and Privacy-Preserving Reconfigurable Authentication Scheme for IoT Devices

Prosanta Gope, *Senior Member, IEEE*, Fei Hongming, and Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) has revolutionized connectivity by enabling a large number of devices to autonomously exchange real-time data over the Internet. However, IoT devices used in public spaces are vulnerable to physical and cloning attacks. To address this issue, researchers have introduced the concept of physical-unclonable functions (PUFs) to enhance security in IoT applications. While PUF-based security solutions typically rely on static challenge-response behavior, many practical applications require dynamic or reconfigurable PUFs. For instance, PUF-based key storage may require updating or revoking secrets, and protection against modeling attacks, where an attacker can derive a PUF model from a set of challenge-response pairs (CRPs) using learning capabilities. In this paper, we introduce LR-OPUF, a reconfigurable one-time PUF, and propose a lightweight and privacy-preserving authentication scheme based on this LR-OPUF foundation. One notable feature of our authentication scheme is that it enables a device to prove its legitimacy to a semi-honest verifier without disclosing the CRPs. Through security and performance analyses, we demonstrate that our approach not only ensures vital security aspects but also exhibits high computational efficiency.**

***Index Terms*—Mutual authentication, IoT, devices, privacy.**

I. Introduction

IOT refers to an interconnected system with a wide range of heterogeneous devices, from high-end smartphones all the way down to simple sensors only capable of gathering and transmitting data. IoT has opened the doors for many new and exciting applications such as smart cities, smart healthcare, smart factories, smart agriculture, etc. The main advantage of IoT devices is their connectivity with relatively modest hardware capabilities. Thus, they play a critical role in automation strategies to control and perform various tasks remotely. Around 75 billion Internet-connected devices are expected by 2025 [1], and currently, this number is 7 billion. The large number of IoT devices producing huge amounts of data leads to many serious security concerns. Moreover, the heterogeneity and resource-constrained nature of most of these devices make IoT even more vulnerable to cyber-attacks. A typical IoT system is composed of numerous low-end embedded devices (for sensing and control) connected to intermediate nodes, i.e., gateways. The gateways usually have a wired connection to the Internet for relaying the data gathered by the IoT devices to application servers in the cloud for data analysis and control. The low cost and simple nature of IoT devices with Internet connectivity can be exploited by adversaries to discover new weaknesses and exploit emerging vulnerabilities reported about specific devices [2], [3]. However, due to the long lifespan and no active maintenance of these devices, most IoT devices are not protected against emerging vulnerabilities. An additional consideration in many IoT-based applications is that the devices often need to be deployed in open and public places. As a result, an adversary can perform physical and cloning attacks to change the settings of the device (to compromise it) or acquire the device's secret credentials, and thus compromise the device and the entire system. Thus, it is important for security solutions for IoT-based applications to consider the physical security of IoT devices. To tackle this challenge, researchers have proposed the use of physical unclonable functions (PUFs) in the existing literature. PUFs serve various purposes, such as hardware authentication and key generation [4], [5]. Due to their resistance against physical and invasive attacks, as well as their ability to securely store keys without explicit storage, PUFs have emerged as strong contenders for enhancing IoT security. However, applying complex logical or arithmetic operations to silicon manufacturing variations is challenging due to the amplification of physical noise. As a result, PUF circuits are typically evaluated in a linear manner, which makes them susceptible to modelling attacks [6], [7], [8], [9].

Received 15 September 2023; revised 3 January 2025; accepted 15 January 2025. Date of publication 29 January 2025; date of current version 10 April 2025. The work of Prosanta Gope was supported by The Royal Society Research Grant under grant RGS R1221183. Biplab Sikdar is supported by A∗STAR, CISCO Systems (USA) Pte. Ltd, in part by the National University of Singapore, and in part by the Cisco-NUS Accelerated Digital Economy Corporate Laboratory under Grant I21001E0002. *(Corresponding authors: Prosanta Gope; Fei Hongming.)*

Prosanta Gope is with the Department of Computer Science, University of Sheffield, S1 4DP Sheffield, U.K. (e-mail: prosanta.nitdgp@gmail.com).

Fei Hongming is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: fei.hongming@u.nus.edu).

Biplab Sikdar is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: bsikdar@nus.edu.sg).

A. Literature Review and Motivation

Since PUFs emerged as a critical primitive for device security, various authentication systems based on PUFs have been developed in the literature. These can be divided into three categories: (i) ideal-PUF-based authentication schemes, (ii) noisy-PUF-based authentication schemes, and (iii) machine-learning (ML) prevention-based authentication schemes. First, we will give some examples of ideal-PUF-based authentication schemes. In 2008, Bringer et al. showed the use of ideal-PUF for designing an RFID authentication scheme [10]. Subsequently, Tuyls

and Skoric proposed a new strong authentication scheme [11]. However, this protocol has been proven to be insecure against various attacks [35]. In follow-up work, a few more interesting PUF-based authentication protocols (e.g., [12]) have been proposed by considering ideal PUF conditions. However, according to [16], most of them are vulnerable to various attacks. Next, we give some examples of PUF-based authentication schemes where noise has been taken into account. In 2011, Brzuska et al. introduced a PUF-based key-exchange (KE) protocol along with two other protocols for oblivious transfer (OT), and bit commitment (BT) [31]. However, the protocols are proven to be insecure against quadratic attacks and under the "Re-Use" model (see Section III-B) of the PUF [35]. Chatterjee et al. [17] presented a public-key-based setup for the secure transfer of data in IoT. The protocol requires an additional component called a security association provider and six rounds of interactions. Subsequently, three more interesting noisy-PUF-based authentication schemes have been proposed [16], [18], [19]. However, they are vulnerable to machine-learning attacks. For instance, in Gope et al.'s scheme [16], if an attacker with machine-learning capabilities have physical access to the device and can obtain a set of CRPs, then he/she will be able to model the PUF by using machine learning algorithms [6], [7], [8], [9]. However, identifying such an attack will be difficult for the verifier. Also, according to [16], the scheme presented in [19] cannot ensure the privacy of the devices.

To deal with the machine learning attacks, Yu et al. [20] introduced two interesting protocols. However, these protocols are not scalable. For instance, the first protocol only allows a limited number of interactions with the server; after that, the device is not allowed to access the server. In the second protocol, for each PUF-embedded device, the server needs to maintain a train-PUF-model [21], which is infeasible for an application with a large number of devices. In addition, none of the protocols has considered the privacy of the devices since, during the authentication process, the device needs to disclose its identity in order to help the server to identify them. Subsequently, Liang et al. [22] proposed another train-PUF-model-based authentication scheme. Unfortunately, the proposed scheme has the same scalability issue as [20]. In addition, our analysis shows that in Liang et al.'s scheme, the server needs to try all possible train-PUF-models in order to identify a particular device. This is because, during authentication, the device does not send any information that can be used by the server to identify the device and uses a specific train-PUF model easily. Therefore, this scheme is not suitable for any application with a large number of devices. Recently, Gu et al. [23] introduced the idea of Fake PUF to mislead the attacker. However, this approach causes a significant overhead at the hardware level. Besides, like [20] and [22] the server in [23] needs to maintain a trained PUF model of the genuine PUF to authenticate a device. In [39], the authors showed a new way to enhance resilience to modelling by adding two flip-flops along with an additional procedure. However, like [23], this approach also causes additional overhead at the hardware level. Besides, like [20], [22], and [23] the protocol designed based on this construction needs to maintain a train-PUF-model at the server end. Barbareschi et al. [40] introduced a new protocol using the concept of chains of CRP (predefined); however, their protocol is vulnerable to device impersonation attacks. In [41], a hybrid PUF structure (called DCH-PUF), along with an authentication protocol, has been proposed to deal with the modelling attacks. However, like [20], [22], and [23], the proposed solution also requires maintaining a PUF-train model at the server's end, which causes the same scalability issue like [20], [22], and [23]. Besides, after thoroughly checking, we found that the proposed authentication scheme is also vulnerable to several attacks, such as replay attacks, man-in-middle attacks, etc. Nevertheless, in all the schemes described above, a server/verifier is allowed to obtain a significant number of CRPs, which can be used for modelling the PUF behaviour or forming a marionette PUF. The symbols and cryptographic functions used in the proposed scheme are defined in Table I.

TABLE I
SYMBOLS AND CRYPTOGRAPHIC FUNCTIONS

Symbol	Definition
RID_T^i	Reference identity of the device T for i-th round
TF_1	Input Transformation Function
TF_2	Output Transformation Function
(S_i)	i-th State stored in NVM
W_i	Masked output of TF_1 for the i-th round
Q_i	Masked output of TF_2 for the i-th round
FE	Fuzzy Extractor
SK	Session key
$h(.)$	Collision-resistant hash function (CRHF)

Our goal and contribution: This paper aims to tackle all of the aforementioned concerns. In order to do that, we first propose a new controlled-PUF construction that we call LR-OPUF (Logically Reconfigurable One-time PUF). Subsequently, we design a *new* reconfigurable authentication scheme using our proposed LR-OPUF construction, where the behavior of the LR-OPUF changes through a reconfiguration process. In a nutshell, this paper's significant contributions can be summed up as follows:

- We propose the LR-OPUF, a reconfigurable Physical Unclonable Function (PUF) construction that leverages both the physical properties of an underlying PUF and state information stored in non-volatile memory (NVM) to ensure its security.
- Our proposal entails a lightweight and privacy-preserving reconfigurable authentication scheme, which has been designed based on the underlying foundation of the LR-OPUF. An important characteristic of the proposed authentication scheme is its ability to prevent a semi-honest server (verifier) from gaining knowledge of the original Challenge-Response Pairs (CRPs) associated with the underlying PUF. Despite this, the scheme enables device authentication without relying on assistance from a trusted third party. To the best of our knowledge, we are the *first* to address this issue, especially in the design of a machine learning resilient authentication scheme.
- We conduct a detailed formal security analysis of our suggested system (constructions) in order to demonstrate that it is secure against many imperative attacks.

II. Preliminaries

A. Physically Unclonable Functions

Physically Unclonable Functions (PUFs) are physical circuits commonly used to map a binary input (a bit-string challenge, denoted as C) to a single or multi-bit output (the response, denoted as Q) [4], [5]. The inherent micro-variance in each integrated circuit produced during the manufacturing process allows for the generation of strong entropy, leading to unpredictable challenge/response behavior that is not based on mathematical functions. A PUF can be represented as $Q \leftarrow \mathrm{P}(C)$. The primary objective of PUFs is to provide robust security for authentication solutions while remaining lightweight, particularly in scenarios where cryptographic resources are limited. PUFs are expected to possess properties such as being unclonable, unpredictable, reliable, and tamper-evident. Generally, PUFs can be classified into two types: strong PUFs and weak PUFs. Strong PUFs have the capability to generate an exponential number of challenge/response pairs (CRPs) and are predominantly used for authentication purposes, employing single-use, known CRPs. However, strong PUFs, which typically consist of linear circuits, are vulnerable to modelling attacks [6]. In contrast, weak PUFs can generate only a few, and often just a single, CRP. Consequently, weak PUFs have been utilized for reliable secret key generation in security protocols. The limited number of CRPs derived from weak PUFs makes modelling attacks that rely on large datasets of known CRPs inapplicable as an attack method. However, other physically invasive and side-channel attacks have been attempted on popular weak PUFs [25].

B. PUF Reconfigurablity

Reconfigurable PUFs are a (relatively) newer development within the lifetime of PUF-related research. The main idea of reconfigurable/dynamic PUF was introduced by Kursawe et al. in [24]. Usually, a PUF exhibits a static challenge/response behavior. However, in many applications, a reconfigurable/dynamic PUF behavior is desirable. For instance, in case of a PUF-enabled reusable/recyclable wireless access token [32], a new user should not be allowed to obtain the security-sensitive information of the previous user of the token. The idea of the reconfigurable PUF can also be useful for the application of PUF-based key-storage [33], [37] and PUF-based cryptographic primitives, where one would need to revoke/update the previous PUF secrets extracted from the PUF. Furthermore, the idea of reconfigurable PUF has been suggested in [36] as a countermeasure against the *PUF Re-use model*. In such scenarios, an adversary is allowed one-time access to the PUF after the execution of each protocol session and can perform CRP-measurement on the PUF (see Section III-B). Recently, the idea of such dynamic/reconfigurable PUF behavior has also been adopted for dealing with machine-learning or modelling attacks [37]. For example, Modifying parameters like the timing of refresh pauses or adjusting the size of the allocated memory block in a DRAM-PUF [34] allows for the creation of novel challenge-response patterns, effectively reconfiguring the system's 'state'. Even though the concept of reconfigurable/dynamic PUF has been considered an effective approach for many PUF-enabled applications, implementing such PUFs in a practical way for designing a cost-effective the security solution is difficult in practice. In order to address this issue, in this paper, we present a Logically Reconfigurable One-Time PUF (LR-OPUF), which is then used to design a privacy-preserving authentication scheme.

C. Modelling Attacks on PUFs

In general, strong PUFs such as arbiter PUFs (APUFs) and XOR arbiter PUFs are susceptible to machine learning attacks that aim to generate a mathematical model of an individual PUF to enable impersonation [25], [26], [27], [28], [29]. These types of attacks generally involve an adversary collecting a large subset of a PUF's possible CRPs by calling repeated measures: $\{(C_1, Q_1), (C_2, Q_2), \ldots, (C_w, Q_w)\}$. Based on this, a predictive model $\hat{m}$ can be developed as an algorithm to estimate an unknown response Q_{w+1} for a new challenge C_{w+1} [9]. The probability that the model correctly predicts the response to an unseen challenge is referred to as accuracy. Once a model has been produced, an adversary can attempt to impersonate a device during an authentication sequence by replying to another device with the correct response to an arbitrary challenge. Rührmair et al. introduced the first machine learning-based modelling attacks (ML-MA) to impact APUFs and XOR APUFs [6] significantly. The three most common and effective types of ML-MA as a result of this work are Logistic Regression (LR), Support Vector Machine (SVM), and Evolutionary Algorithms (EA) such as Evolution Strategies (ES). Probably Approximately Correct (PAC) [29] machine learning and Artificial Neural Network (ANN) techniques [27] have also been proven to be effective in attacking these PUF implementations. To base our assumptions on the security of One-Time PUFs, we first mention the following proven attacks on non-reconfigurable PUFs. Logistic regression is a well-investigated supervised machine learning algorithm used to predict the probability of a target variable, in the case of PUFs: unknown responses. The model predicts the likelihood of A PUF output bit is either a one or a zero. More specifically, each challenge C is assigned a probability $p(C, l|\vec{w})$ that it generates an output $l \in \{0, 1\}$. The vector $\vec{w}$ encodes the related internal parameters, e.g., run-time delays of a specific PUF. Rührmair et al. proved a very successful Logistic Regression attack (LR-RPROP) on the XOR-APUF, the largest being a 5-XOR, 128 stage APUF with a 500,000 CRP dataset in [6]. In [14], Wisiol et al. exploited the local minima produced by the input transformation of Lightweight Secure PUFs to model them successfully. They also provided a divide-and-conquer modelling strategy to attack the Interpose PUF (iPUF) in [15], involves a Linear Regression attack on its lower layer with randomly interposed bits and uses this to model the upper layer further and thus, the entire iPUF.

III. Proposed Scheme

Within this section, we initially introduce our LR-OPUF design, which is the foundation of the proposed reconfigurable authentication scheme. Subsequently, we define the adversary

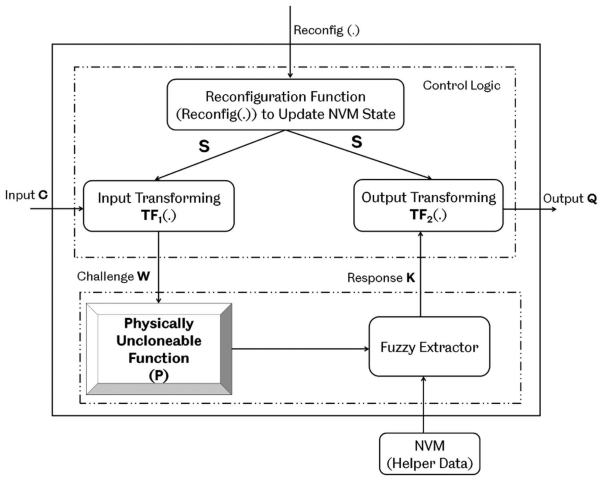


Fig. 1. LR-OPUF construction.

model along with the security goals. Finally, we present our reconfigurable authentication scheme that prioritizes lightweight implementation and privacy preservation. The scheme comprises two distinct phases.

A. Proposed LR-OPUF Construction

The term "*One-Time PUF* (OPUF)" refers to a system for changing the PUF configuration/settings after each authentication session, which can be considered as a variant of a reconfigurable PUF. An OPUF is designed to provide both forward and backward unpredictability. Forward unpredictability ensures that the measured responses of the PUF before the resetting event are considered invalid. On the other hand, backward unpredictability ensures that even if an adversary gains access to the current state of the PUF, they cannot estimate the PUF's behavior prior to the reconfiguration. Now, like the other reconfigurable PUFs, implementing OPUFs in a practical way for designing a cost-effective security solution is difficult in practice. Therefore, we now present our logically reconfigurable OPUF (LR-OPUF) construction, depicted in Fig. 1, as the fundamental basis of our proposed authentication scheme. It is important to note that LR-OPUF can be viewed as Controlled PUFs, which aim to conceal the challenge/response behavior of the underlying PUF to enhance security against modelling attacks. Our construction consists of two primary components: a control logic circuit and a conventional strong PUF. The control logic circuit encompasses three dedicated functions: Reconfig$(\cdot)$, TF_1, and TF_2. To modify the output behavior of the LR-OPUF, the circuit employs the Reconfig$(\cdot)$ function to reconfigure the state S to a new independent random state $S' \leftarrow$ Reconfig$(\cdot)$ when necessary. The updated state S' is securely stored in the non-volatile memory (NVM) of the device. The input-transforming function, TF_1, maps/transforms the input challenge C into the challenge W using the operation $W \leftarrow h(S||C)$, where $h(\cdot)$ is a one-way collision-resistant hash function. Subsequently, the PUF output R is obtained by utilizing W as the challenge, denoted as $R \leftarrow P(W)$. Next, the error correction on the noisy PUF output R is performed by using a *Fuzzy Extractor* (FE) and the helper data (hd) stored in the NVM, i.e., $K \leftarrow FE.Rec(R||hd)$. Hereafter, the output-transforming function TF_2 is used to transform the *Fuzzy Extractor* output K into the final masked output Q, i.e., $Q \leftarrow h(S||K)$. Roughly speaking, the transformation functions TF_1 and TF_2 can be regarded as masking functions, where TF_1 masks the PUF input and TF_2 masks the PUF output. Upon the completion of each authentication phase, the Reconfig$(\cdot)$ function is invoked to update and reconfigure the state of the PUF, denoted as $S' \leftarrow h(S)$. The functions Reconfig$(\cdot)$, TF_1, and TF_2 are publicly known, such as widely recognized hash functions. It is important to note that an adversary $\mathcal{A}$ lacks the capability to manipulate or modify the state to a value of their choosing.

B. Adversary Model

We take into account the following adversary capabilities in our proposed LR-OPUF-based authentication technique. First, an adversary can intercept the communication channel between the device and the server under our proposed system. The adversary can also try to alter/block the exchanged messages. In addition to that, an adversary can also try to mount cloning and physical attacks on the PUF. In our proposed scheme, we have also taken into account the adversary's potential to attempt machine learning modelling attacks (as described in Section II.C). For that, we first consider the *Re-Use Model* of [35], where an adversary may have repeated physical access (e.g., through the setup of a fake terminal) to the the device between different key exchange sessions and try to obtain a considerable numbers of $\{C, Q\}$ pairs (input/output pairs of the LR-OPUF) in order to model the behaviour of the LR-OPUF. Moreover, our adversary model considers the authentication server (operated by commercial entities) to be semi-honest and allows for the acquisition of a considerable number of session messages for authentication purposes. In such cases, consider a multi-server scenario, if the PUF-enabled device (such as a wireless token received from a trusted authority) is registered with multiple servers (for different services) and one of the servers is malicious. It should be noted that we consider different states, and registration phases are launched between the device and different servers. After obtaining a considerable number of CRPs, the malicious server can construct a train PUF model that can be used to impersonate a legitimate device with other entities. However, neither the victim device nor the victim server will be able to detect such malicious authentication attempts. Additionally, we also allow multiple malicious servers to collude with each other to obtain a large number of CRPs or input/output pairs and eventually model the LR-OPUF behaviour within a relatively shorter time span. This model can then be used to impersonate a legitimate device. Section IV delves into the adversary concept in further depth.

C. Security Objectives

Our primary focus revolves around four crucial security requirements: *forward unpredictability*, *backward unpredictability*, *resilience against modelling attacks*, and *privacy of IoT devices against eavesdropping or tractability attacks*.

- *Forward unpredictability:* In terms of forward unpredictability, it is of utmost importance that an adversary, denoted as $\mathcal{A}$, is unable to predict the PUF output (R), final masked output (Q), and current state (S) for any new input C. This holds true even if $\mathcal{A}$ has access to an adaptively chosen set of input/output pairs of the LR-OPUF for any previous state.
- *Backward unpredictability:* In the context of backward unpredictability, it is necessary that $\mathcal{A}$ cannot predict the PUF output (R), final masked output (Q), and current state (S) based on any previous state (prior to reconfiguration) when given a new challenge C. This requirement remains even if the adversary has the capability to adaptively acquire challenge/response pairs or input/output pairs for the current state.
- *Privacy:* Given that IoT devices frequently deal with sensitive user information, it is crucial to ensure that communication between these devices and the server maintains a high level of anonymity and confidentiality. It is imperative to prevent eavesdroppers from being able to identify the device or track its movements.
- *Non-resettability:* The adversary $\mathcal{A}$ should face significant challenges in attempting to manipulate the state of the LR-OPUF to achieve a specific desired value. The system should be designed in a way that makes it practically impossible for $\mathcal{A}$ to successfully set the LR-OPUF's state to the desired value, ensuring the integrity and security of the system.
- *Resilience against modelling attacks:* To carry out modelling attacks against a Physical Unclonable Function (PUF), adversaries typically require access to a substantial number of Challenge-Response Pairs (CRPs). Therefore, to guarantee security against such modelling attacks, it is crucial to prevent any adversary, including the semi-honest server, denoted as $\mathcal{A}$, from successfully modelling the behavior of the LR-OPUF. This holds true even if $\mathcal{A}$ manages to obtain a significant quantity of CRPs or input/output pairs. Safeguarding against modelling attacks is essential to maintain the robustness and integrity of the LR-OPUF system.
- *Other objectives:* In addition to addressing privacy concerns, forward unpredictability, backward unpredictability, and modelling attacks, the proposed scheme takes into account several other crucial security properties. These include protection against impersonation or forgery attacks, as well as defenses against replay attacks and other relevant security threats. By considering these additional security properties, the scheme aims to provide comprehensive protection and ensure the integrity and authenticity of the system.

D. Assumptions

The initial assumption made is that an LR-OPUF instance remains valid solely for a specific session during the execution of the authentication protocol, and it cannot be utilized in any other session. This restriction ensures that the LR-OPUF instance is securely bound to its intended session and prevents its unauthorized reuse in subsequent sessions. Here, we argue that the forward and backward unpredictability of a state-reconfigurable LR-OPUF provides significant PUF-level security against machine learning attacks. Even if repeated measures are utilized to generate a large enough CRP dataset to model a single state of an LR-OPUF, by this time a new authentication session will have opened and thus a new LR-OPUF state would have been adopted. The adversary's model will not have sufficient accuracy to predict unknown responses to new challenges for the new LR-OPUF state. Thus, the adversary would have to attempt a modelling attack again on the new state, thereby continuing the cycle. In our proposed authentication scheme, we assume that all actions taken within the setup (i.e., enrollment) phase is inaccessible to all the adversaries (described in Section III-A and Section IV). As a result, every adversary makes an effort at an attack during the authentication phase. We suppose that an adversary has gained physical access to the device between the execution of two authentication phase sessions. Thus, the adversary has the potential to gain access to the LR-OPUF's interface and employ brute force techniques to query the device with various challenges. The adversary aims to construct their own Challenge-Response Pairs (CRPs) dataset to train a machine learning model for a potential modelling attack by monitoring and collecting the corresponding responses. However, it is important to note that any attempt by the adversary to tamper with the LR-OPUF physically would disrupt its functionality, rendering it useless. Therefore, the integrity of the LR-OPUF is maintained, even if the adversary has physical access to the device in their efforts to obtain CRPs. Now, if we consider the proposed protocol to be used in a scenario where a device registers with multiple verifiers for different services. Specifically, we here would like to emphasise that the device must run an individual setup phase with each individual server, which means different session keys and stored data are used for different servers. In this way, one semi-honest server cannot directly compromise the credential secrets used by other servers.

E. Proposed Lightweight and Privacy-Preserving Reconfigurable Authentication Scheme Using LR-OPUF

The proposed reconfigurable authentication scheme can be divided into two distinct phases: the setup phase and the reconfigurable authentication phase. The setup phase of our proposed scheme is executed between a device and the server in a secure environment (through a secure channel). On the other hand, the authentication phase will take place in an insecure environment (insecure public/wireless channel).

1) Setup Phase: In this phase (shown in Fig. 2), first, a device randomly generates a new independent state S_i and generates a challenge C_i. Then the device computes $W_i = \mathrm{TF}_1(C_i) \Rightarrow h(C_i||S_i)$ (for *masking/transforming* the input), $R_i = P_T(W_i)$, $(K_i, hd_i) = \mathrm{FE.Gen}(R_i)$, and $Q_i = \mathrm{TF}_2(K_i) = h(S_i||K_i)$, where $Q_i = \{Q_i^1||Q_i^2\}$and $i = 0$. Finally, the device constructs a message. $\mathrm{Set}_1 : \{ID_T, (K_i, Q_i)\}$ and sends it to the server for enrollment, where the parameter ID_T denotes the identity of the device. On receiving Set_1, the server generates a

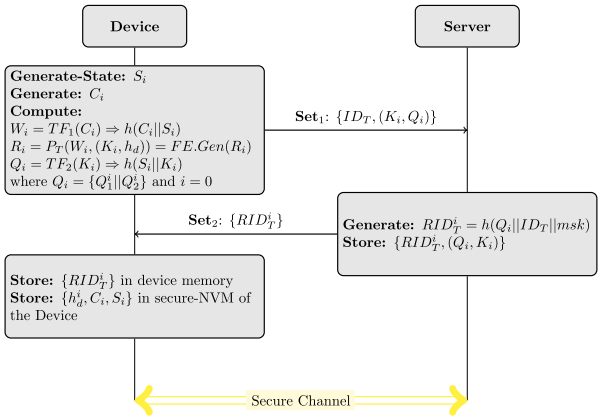


Fig. 2. Setup phase of proposed LR-OPUF-based authentication scheme.

reference identity for the *i*-th session of authentication, $RID_T^i = h(Q_i||ID_T||msk)$, to uniquely identify the device and sends RID_T^i to the device through a secure channel, where msk denotes the secret master key of the server. Hereafter, the server stores $\{RID_T^i, (Q_i, K_i)\}$ for the *i*-th session of authentication. Next, upon receiving the reference identity RID_T^i, the device stores $\{hd_i, C_i, S_i\}$ in its secure NVM andRID_T^i in its main memory.

2) Reconfigureable Authentication Phase: This phase of the proposed scheme consists of *four* steps, described as follows.

Step #1: For the *i*-th session of execution of the authentication protocol, the device first loads the state S_i, the helper-data hd_i, and the challenge C_i from its NVM. Next, the device selects its reference identity RID_T^i and also generates a random number N_t. Hereafter, the device computes $W_i = \mathrm{TF}_1(C_i) \Rightarrow h(C_i||S_i)$ and subsequently composes a message $\mathrm{MSG}_1 : \{RID_T^i, N_t\}$ and sends MSG_1 to the server through an insecure public channel.

Step #2: Upon receiving message MSG_1 from the device, the server first checks its database and tries to find reference id RID_T^i in it. If the server is unable to locate RID_T^i in its database, then it aborts the execution of the protocol, and the device has to retry with a valid security credential (as discussed at the end of this section). Otherwise, the server reads (Q_i, K_i) from its database and loads them into its main memory, where $Q_i = \{Q_i^1||Q_i^2\}$. After that, the server generates a nonce N_s and computes $N_s^* = K_i \oplus N_s$, $Q_i^{1*} = Q_i^1 \oplus K_i$, and $\Pi_1 = h(N_s||K_i||Q_i^{1*}||N_t)$ and finally composes message $\mathrm{MSG}_2 : \{Q_i^{1*}, N_s^*, \Pi_1\}$ and sends it (MSG_2) to the device.

Step #3: When the device receives message MSG_2, it first extracts the PUF output $R_i' = P_T(W_i)$ and then calculates $K_i = \mathrm{FE.Rec}(R_i', hd_i)$, $N_s = K_i \oplus N_s^*$, and $Q_i^1 = Q_i^{1*} \oplus K_i$. After that, the device computes $\{Q_i^{1\Psi}||Q_i^{2\Psi}\} = \mathrm{TF}_2(K_i) \Rightarrow h(S_i||R_i)$ and compares Q_i^1 with $Q_i^{1\Psi}$. If $Q_i^1 = Q_i^{1\Psi}$, then the device validates the parameter Π_1. If the validation is successful, then the device computes $Q^* = Q_i^{2\Psi} \oplus K_i$, $C_{i+1} = h(C_i||N_s||N_t)$, and the session key $SK = h(N_t||N_s||Q_i^{\Psi})$, where $Q_i^{\Psi}[\{Q_i^{1\Psi}||Q_i^{2\Psi}\}] = Q_i[\{Q_i^1||Q_i^2\}]$ (i.e., $Q_i^{1\Psi} = Q_i^1$ and $Q_i^{2\Psi} = Q_i^2$). Hereafter, the device reconfigures its state to a new independent state by invoking the Reconfig$(\cdot)$ function, i.e., $S_{i+1} = \mathrm{Reconfig}(S_i) \Rightarrow h(S_i||R_i')$, and computes $W_{i+1} = \mathrm{TF}_1(C_{i+1}) \Rightarrow h(C_{i+1}||S_{i+1})$. Then, the device extracts the PUF output $R_{i+1} = P_T(W_{i+1})$ and also computes $(K_{i+1}, hd_{i+1}) = \mathrm{FE.Gen}(R_{i+1})$ for the *(i*+1)-th session of secure interaction. Subsequently, the device calculates $Q_{i+1} = \mathrm{TF}_2(K_{i+1}) \Rightarrow h(S_{i+1}||K_{i+1})$ for *masking/transforming* the output of the FE and then computes $\Delta = \mathrm{Enc}[Q_{i+1}||K_{i+1}]$, $\Pi_2 = h(\Delta||K_i||Q^*||N_s)$, $RID_{i+1}^T = h(RID_T^i||Q_{i+1})$, and $K_{i+1} = h(K_i||Q_{i+1})$. At the end, the device forms the message $\mathrm{MSG}_3 : \{Q^*, \Pi_2, \Delta\}$ and sends the message to the server. Subsequently, the device stores RID_{i+1}^T in its main memory and updates the key K_i with the new key K_{i+1} and challenge C_i with the new challenge C_{i+1}.

Step #4: When the server receives message MSG_3, it first computes $Q_i^{2\Psi} = Q^* \oplus K_i$ and compares $Q_i^{2\Psi}$ with Q_i^2. If the verification is successful, then the server also verifies the key-hash parameter Π_2. If the server successfully verifies both parameters, it implies that the server authenticates the device as a legitimate entity. Next, the server calculates the session key $\mathrm{SK} = h(N_t||N_s||Q_i)$, where $Q_i^{\Psi}[\{Q_i^{1\Psi}||Q_i^{2\Psi}\}] = Q_i[\{Q_i^1||Q_i^2\}]$, and subsequently decrypts Δ and obtains mr$Q_{i+1}||K_{i+1}$. After that, the server computes $RID_{i+1}^T = h(RID_T^i||Q_{i+1})$ and K_{i+1}=$h(K_i\|Q_{i+1})$, and at the end it replaces the key K_i with the new key K_{i+1} and stores $\{RID_{i+1}^T, (Q_{i+1}, K_{i+1})\}$. Details of the proposed reconfigurable authentication process are shown in Fig. 3. The successful completion of all the aforementioned steps in the authentication protocol confirms the mutual authentication between the server and the device T, resulting in the establishment of a shared session key SK. Any failure in the verification process leads to the termination of the protocol execution.

To address any potential desynchronization between the device and the server, we propose incorporating additional measures into the protocol. In this regard, apart from K_i, Q_i, the device also needs to generate a set of EM (emergency) challenges $C_{\mathrm{EM}} = \{C_{\mathrm{EM}}^1, \ldots, C_{\mathrm{EM}}^n\}$ during the execution of the setup phase. Then, using the same process as for K_i, Q_i, the device generates $[(K_{\mathrm{EM}}, Q_{\mathrm{EM}})] = \{(K_{\mathrm{EM}}^1, Q_{\mathrm{EM}}^1), \ldots, (K_{\mathrm{EM}}^n, Q_{\mathrm{EM}}^n)$ to enhance the security of the stored emergency data and mitigate the risk identified by potential attackers accessing this data, we propose encrypting the stored $EID_{\mathrm{EM}}, K_{\mathrm{EM}}, Q_{\mathrm{EM}}$ using Authenticated Encryption with Associated Data (AEAD). The encryption key used is K_i, and the associated data (AAD) includes C_{EM} and hd_{EM}. Before storing the data in its non-volatile memory (NVM) and sending it to the server through the secure channel, the device encrypts $EID_{\mathrm{EM}}, C_{\mathrm{EM}}K_{\mathrm{EM}}, Q_{\mathrm{EM}}$ using this method.

Next, the server generates a set of unique emergency identities $EID_{\mathrm{EM}} = \{EID_{\mathrm{EM}}^1, \ldots, EID_{\mathrm{EM}}^n\}$ and securely transmits EID_{EM} to the device through a secure channel. The server also stores an encrypted copy of $EID_{\mathrm{EM}}, K_{\mathrm{EM}}, Q_{\mathrm{EM}}$ in its database using the same AEAD encryption method.

In case of loss of synchronization, both the server and the device need to use a set $\{EID_{\mathrm{EM}}^j, (K_{\mathrm{EM}}^j, Q_{\mathrm{EM}}^j)\}$ from the encrypted $\{EID_{\mathrm{EM}}, (K_{\mathrm{EM}}, Q_{\mathrm{EM}})\}$. After a set of data is utilized,

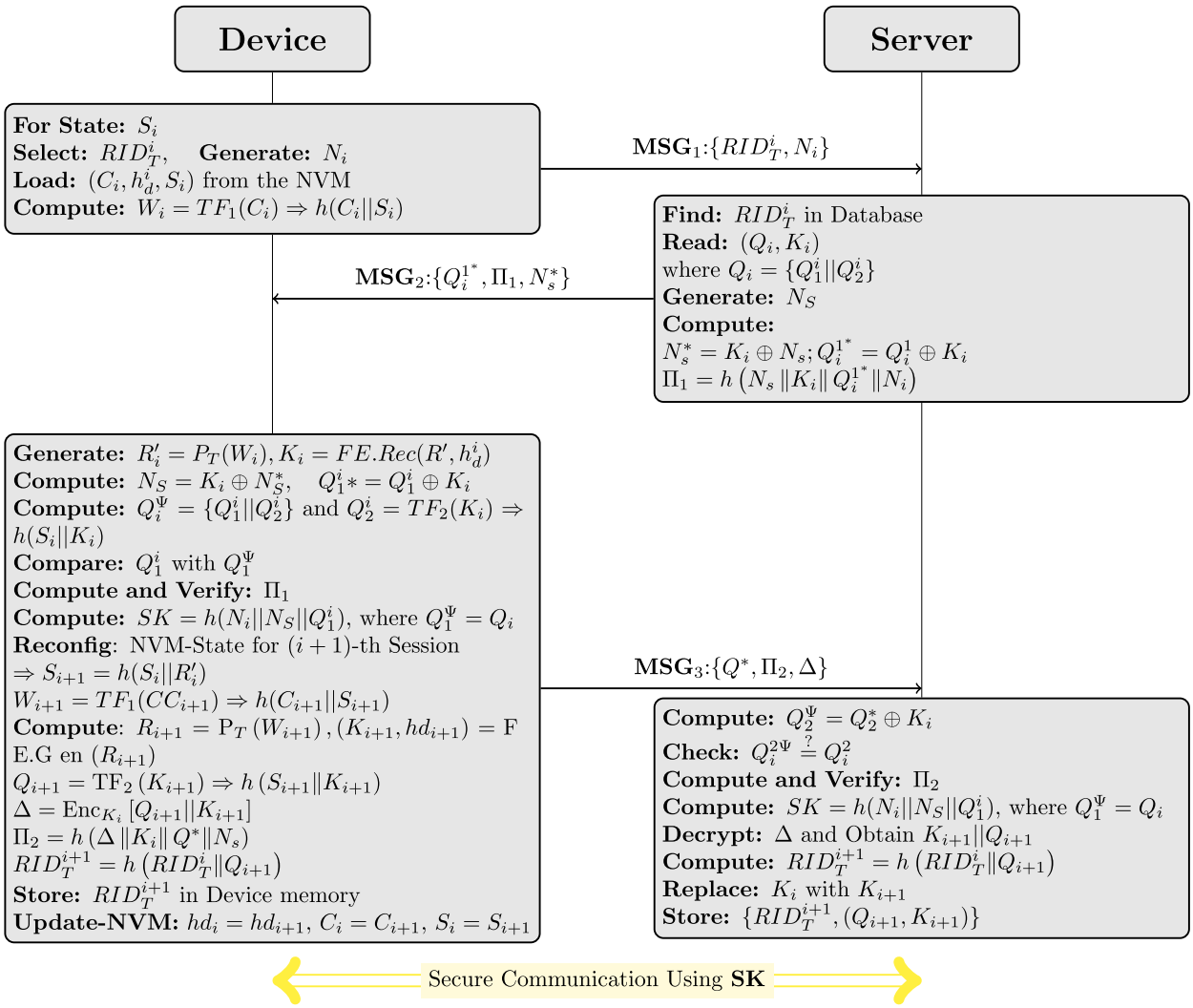


Fig. 3. Proposed LR-OPUF-based modelling-attack resilient reconfigurable authentication scheme.

it must be removed from both the device and the server, necessitating the repetition of the generation and encryption process.

By adopting this approach—where the emergency data is securely encrypted using AEAD with key K_i and associated data C_{EM} and hd_{EM}—the protocol effectively handles desynchronization between the device and server without compromising privacy or security. Even if an attacker gains access to the stored encrypted emergency data, they cannot decrypt or manipulate the data without K_i and the correct associated data. By discarding and regenerating sets of encrypted data, the protocol maintains its ability to operate smoothly and securely, even when synchronization issues arise.

IV. Security Analysis

In this section, we evaluate our proposed scheme in terms of the significant security requirements discussed in Section III-C. To accomplish this, we begin by introducing our formal adversarial model.

A. Security Primitives

Definition 1: A physical unclonable function (PUF) is said to be $(q_\alpha, q_\beta, \varepsilon_1)$-securely forward and backward unpredictable if:

- *Forward unpredictability:* Given up to q_α challenge-response pairs (C_i, R_i), where C_i is the challenge and $R_i = \text{PUF}(C_i)$ is the response, it is computationally hard to predict the response R_j for any new challenge C_j, with advantage at most ε_1.
- *Backward unpredictability:*After observing up to q_β new challenge-response pairs (C_k, R_k), it remains computationally infeasible to find a valid challenge-response pair from the past interactions with advantage at most ε_1.

Definition 2: A hash function $h(\cdot) : \{0,1\}^* \to \{0,1\}^n$ is said to be an ε_2-secure collision-resistant hash function (CRHF) if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability of finding two distinct inputs $x \neq x'$ such that $H(x) = H(x')$ is bounded by ε_2. Formally,

$$\Pr[\mathcal{A}(1^\lambda) \to (x, x') \text{ such that } x \neq x' \text{ and } H(x) = H(x')] \leq \varepsilon_2 \quad (1)$$

B. Adversarial Model

Assume that, there are n IoT devices $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ communicates with a server $\mathcal{S}$. The server executes $\mathrm{SetupT}(1^\gamma)$ for initializing each device, which is basically a setup algorithm that outputs ψ and secret key $\mathcal{K}$. Both the devices in $\mathcal{T}$ and the server $\mathcal{S}$ participate in the authentication phase of the proposed scheme and try to validate each other through an insecure channel. At the end of the authentication phase, both the devices in set $\mathcal{T}$ and the server $\mathcal{S}$ produce an output of 1 or 0, representing "Accept" or "Reject" as the outcome of the authentication phase. Each communication between a device $T \in \mathcal{T}$ and the server $\mathcal{S}$ is considered a communication session, identified by a unique session identifier sid. A session is labeled as a "matching session" if the messages exchanged between $\mathcal{S}$ and T are honestly transmitted. For the protocol to be deemed correct, both the server $\mathcal{S}$ and the device T must accept the session in the presence of a matching session. To formally analyze the security of the mutual authentication protocol $\mathcal{P}$, an adversary $\mathcal{A}$ engages in a security game with a challenger $\mathcal{C}$.

The experiment output Φ to indicate whether the challenger $\mathcal{C}$ accepts the session or not.

$\mathrm{Expr}^{\mathrm{Sec}}_{\mathcal{P},\mathcal{A}}(\gamma)$:

1) $(\psi, \mathcal{K}) \xleftarrow{Random} \mathrm{Setup}(1^\gamma)$;
2) $(\mathrm{sid}^*, \mathcal{T}) \xleftarrow{Random} \mathcal{A}_1^{\mathrm{Launch},\mathrm{SendS},\mathrm{Send}\mathcal{T},\mathrm{Outcome},\mathrm{Reveal}}(\psi, \mathcal{S}, \mathcal{T})$;
3) $\Phi := \mathrm{Outcome}(\mathrm{sid}^*, \varpi)$;
4) Output Φ.

When the adversary $\mathcal{A}$ receives $(\psi, \mathcal{S}, \mathcal{T})$, she can issue the following oracle queries:

-$\mathrm{Launch}(1^\gamma)$: A new session has been launched by $\mathcal{S}$

-$\mathrm{Send}\mathcal{S}(M)$ A random message M is sent to $\mathcal{S}$.

-$\mathrm{Send}\mathcal{T}(T_j, M)$: An arbitrary message M has been sent to the device $T_j \in \mathcal{T}$.

-Result(ϖ, sid): Output whether the sid of ϖ is accepted or not where $\varpi \in \{\mathcal{S}, \mathcal{T}\}$.

-Reveal(T_j): Output all the information stored in the memory of the IoT device T_j.

The advantage of the adversary $\mathcal{A}$ against $\mathcal{P}$ can be defined as $\mathrm{Adv}^{\mathrm{Sec}}_{\mathcal{P},\mathcal{A}}(\gamma)$, which represents the probability that $\mathrm{Expr}^{\mathrm{Sec}}_{\mathcal{P},\mathcal{A}}(\gamma)$ outputs 1 when sid^* of ϖ does not have a corresponding session.

Definition 3: The security of a protocol $\mathcal{P}$ against man-in-the-middle attacks can be defined as follows: For any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\mathrm{Sec}}_{\mathcal{P},\mathcal{A}}(\gamma)$ is negligible in γ, where γ is a security parameter sufficiently large.

Theorem 1 (Security): Assuming that the LR-OPUF instance $\mathcal{LR}\text{-}\mathcal{OPUF}^*$ is a $(q_\alpha, q_\beta, \varepsilon_1)$-secure forward and backward unpredictable PUF, and $h(\cdot)$ is an ε_2-secure collision-resistant hash function (CRHF) (undefined), the proposed authentication protocol $\mathcal{P}$ can be deemed secure against man-in-the-middle attacks.

Proof: The attacker's objective is to manipulate the security game by attempting to persuade either the device $T_j \in \mathcal{T}$ or the server $\mathcal{S}$ to accept a session that does not correspond to the intended session. To prove the security of our protocol, we adopt a security game approach where we gradually replace the protocol's communications with random strings. The adversary is considered successful if she can distinguish between real execution and random string execution, and successfully modify the messages exchanged between the device and the server to make them accept the non-matching session. We proceed with the game transformations, denoting $\mathcal{G}_i$ as the adversary's advantage in winning Game i.

- *Game 0:* The primary game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ is conducted without making any modifications to the protocol.
- *Game 1:* During the execution of the authentication session between the server $\mathcal{S}$ and a specific device $T_j \in \mathcal{T}$, we introduce alterations to the parameters of the LR-OPUF. Subsequently, we evaluate the output of the LR-OPUF in T_j. Given the assumption that the LR-OPUF is a $(q_\alpha, q_\beta, \varepsilon_1)$-secure backward and forward unpredictable PUF (undefined), it can be concluded that the output of the LR-OPUF satisfies the min-entropy property, ensuring the absence of correlation between each output. Based on this property, if an adversary issues a Reveal query and obtains the stored information from the LR-OPUF's memory, the resulting output will exhibit correlation. This observation indicates that the game transformation from Game 1 remains unaffected. Furthermore, if the adversary $\mathcal{A}$ fails to successfully impersonate a legitimate T_j to the server $\mathcal{C}$, the challenger $\mathcal{C}$ aborts the game.
- *Game 2:* At this stage, we can make the assumption that the adversary is capable of establishing a maximum of L sessions in the game. For each session, denoted by $1 \leq m \leq L$, we will modify and subsequently assess the variables associated with the session between the device T_j and the server $\mathcal{S}$ through a series of distinct games.
  - *Game 2-m-1:* During the m-th session, the challenger $\mathcal{C}$ examines the output of the LR-OPUF implemented in device T_j. If it is determined that the LR-OPUF's output lacks sufficient entropy, the challenger $\mathcal{C}$ will abort the game.
  - *Game 2-m-2:* In the context of entity authentication, the challenger $\mathcal{C}$ introduces modifications to the outputs of the collision-resistant hash function $h(\cdot)$. These alterations consist of substituting the hash function outputs with randomly generated strings of the same length. The assumption is that the underlying hash function is secure. If an adversary is capable of distinguishing between the output generated by the function and the output of the random string, it implies that the adversary possesses the ability to break the underlying one-way key-hashed function.
  - *Game 2-m-3:* Next, $\mathcal{C}$ alters the pseudorandom string of $N_s^* = N_s \oplus K_i$ and $Q_i^{1*} = K_i \oplus Q_i^1$ with an arbitrarily generated output-strings. However, since K_i is not accessible to the adversary, as a result, she won't be able to tell the difference between these strings and truly random strings.
  - *Game 2-m-4:* $\mathcal{C}$ substitutes for the pseudorandom string of $Q^* = Q_i^{2\Psi} \oplus K_i$, $\Delta = \mathrm{Enc}[Q_{i+1}||W_{i+1}]$, and $\Pi_2 = h(\Delta||K_i||Q^*||N_s))$ using identically sized strings created at random. Since the adversary has no access to

K_i (as a pseudorandom output obtained from the LR-OPUF), The adversary will be unable to tell the difference between these and truly random strings

We'll change the messages for the IoT device T_j in this section. If the attacker can distinguish the arbitrary strings from real messages/outputs, we claim he/she has won the game. Starting with the initial call of device T_j, we commence the game transformation. After that, we gradually transition from Game 2-m-1 to Game 2-m -4in terms of communication messages. We'll move on to the next step after these transformations are complete. We show that the adversary's advantage over the authentication protocol may be minimized to negligible levels using these game transformations, as indicated in Lemmas 1, 2, 3, 4, and 5.

Lemma 1: If the number of IoT devices is denoted as n, then we can express $\mathcal{G}0$ as n multiplied by $\mathcal{G}1$.

Proof: Due to the presence of n devices, the challenger $\mathcal{C}$ has a reliable probability of $1/n$ to correctly guess the associated session. ■

Lemma 2: If LR-OPUF$_{T_j}$ is a $(q_\alpha, q_\beta, \varepsilon_1)$-secure backward and forward unpredictable PUF, then $\mathcal{G}_1 = \mathcal{G}_{2-m-1}$ and $\mathcal{G}_{2-(m-1)-4} = \mathcal{G}_{2-m-1}$, for any $2 \leq m \leq L$.

Proof: The LR-OPUF attached with device T_j is a $(q_\alpha, q_\beta, \varepsilon_1)$-secure forward and backward unpredictable PUF and The PUF's min-entropy is higher than χ. Furthermore, even if the PUF's input is revealed, the output generated from it preserves a sufficient min-entropy quality, and the outputs are thus uncorrelated. Next, if an adversary makes the Reveal query and obtains the stored information from the OPUF's memory, then, because the games in $\mathcal{G}_1$, $\mathcal{G}_{2-m-1}$ and $\mathcal{G}_{2-(m-1)-4}$ are based on the above condition, the gap between them is bounded by ε_1. Therefore, we can write $|\mathcal{G}_1 - \mathcal{G}_{2-m-1}| \leq \varepsilon_1$ and $|\mathcal{G}_{2-(m-1)-4} - \mathcal{G}_{2-m-1}| \leq \varepsilon_1$. This signifies that the game transformations have no effect. ■

Lemma 3: Let $\mathrm{Advr}^{\mathrm{CRHF}}_{\langle(\cdot),\mathcal{B}}(k)$ denote the advantage of $\mathcal{B}$ to break the security of the CRHF $h(\cdot)$. Then, $\forall\, 1 \leq m \leq L$, we have $|\mathcal{G}_{2-m-1} - \mathcal{G}_{2-m-2}| \leq \mathrm{Advr}^{\mathrm{CRHF}}_{\langle(\cdot),\mathcal{B}}(k)$.

Proof: Let us construct an algorithm $\mathcal{B}$ that compromises the security of the Collision-Resistant Hash Function (CRHF) $h(\cdot)$. The purpose of $\mathcal{B}$ is to generate all the necessary security credentials and simulate our protocol, mimicking all aspects except for the i-th session (the current session). Within this algorithm, $\mathcal{B}$ has access to the CRHF $h(\cdot)$. If the adversary initiates the i-th session, $\mathcal{B}$ generates and transmits the randomly distributed challenge $N_s, \xleftarrow{U}, 0, 1^k$ as the output of the server. When $\mathcal{A}$ sends $N_s^{\#}$ to the device, $\mathcal{B}$ proceeds with the computations according to the protocol specification. Instead of performing the usual $h(\cdot)$ computation, $\mathcal{B}$ directs $N_s^{\#}$ to the oracle. Upon receipt of Π_2, $\mathcal{B}$ produces the output Q, Π_2, Δ as the response from the device.

Upon receiving $Q^{\#}, \Pi_2^{\#}, \Delta^{\#}$ from the adversary, $\mathcal{B}$ forwards $N_s^{\#}$ to the oracle and retrieves Π_2, which is subsequently utilized to verify the authenticity of the device. This simulation corresponds to either Game 2-m-1 if $\mathcal{B}$ has access to the CRHF, or Game 2-m-2 if the oracle query made by $\mathcal{B}$ is completely random, resulting in an identical distribution.

Thus, we may compose $|\mathcal{G}_{2-m-1} - \mathcal{G}_{2-m-2}| \leq \mathrm{Advr}^{\mathrm{CRHF}}_{\mathrm{h}(\cdot),\mathcal{B}}$.■

Lemma 4: $\forall\, 1 \leq m \leq L$, $|\mathcal{G}_{2-m-2} - \mathcal{G}_{2-m-3}| \leq \mathrm{Advr}^{\mathrm{CRHF}}_{\mathrm{h}(\cdot),\mathcal{B}}(k)$.

Proof: This lemma's proof is similar to that of Lemma 3. ■

Lemma 5: $\forall\, 1 \leq m \leq L$, *we have* $\mathcal{G}_{2-m-2} = \mathcal{G}_{2-m-3} = \mathcal{G}_{2-m-4}$.

Proof: The LR-OPUF and CRHF $h(\cdot)$ are replaced with the genuinely random function in the three games analyses in this lemma. Thus, K_i/K_{i+1} and $h(\Delta||K_i||Q^*||N_s)$ are used as effective one-time pads to encode $Q_i^{2\Psi}$, $Q_i^{1*} = Q_i^1 \oplus K_i$, and $\Delta = \mathrm{Enc}[Q_{i+1}||W_{i+1}||K_{i+1}]$, and support the integrity of $Q_i^{2\Psi}$ and Δ, respectively. As a result, no adversary will be able to tell the difference $\Pi_2 = h(\Delta||K_i||Q^*||N_s)$, Δ and Q^* from a randomly chosen string. ■

Theorem 2: Assume that the LR-OPUF instance $\mathcal{LR}$-$\mathcal{OPUF}^* \leftarrow$ LR-OPUF is a $(q_\alpha, q_\beta, \varepsilon_1)$-secure backward and forward unpredictable PUF and let $h(\cdot)$ be an ε_2-secure CRHF. The indistinguishability-based privacy property is then satisfied by our protocol $\mathcal{P}$.

Proof: This theorem's proof is identical to Theorem 1, in which we demonstrated that the suggested authentication protocol is secure against *man-in-the-middle attacks* Furthermore, it should be noted that the communication messages for devices $T0^*$ and $T1^*$ are continuously altered during the game transformation described in the proof of Theorem 1. This constant alteration ensures that the entire transcript appears as a random string, thereby preventing any disclosure of information about the challenger's coin. It is important to emphasize that all identity-related memory parameters, including the reference id RID_T^i and the emergency identities $EID_{EM} = (EID_{EM}^1, \ldots, EID_{EM}^n)$, are randomly generated and can only be used once for each pair. As a result, the likelihood that the challenger will be able to identify T_0^* and T_1^* in such a way that the game transformation is completed in a polynomial time is $1/n^2$, where n is the number of devices in the network. To put it another way, no adversary can tell the difference between messages from devices T_0^* and T_1^* with a probability greater than $1/n^2$. As a result, we may argue that the proposed approach can provide privacy based on indistinguishability.■

Theorem 3: Assume that an LR-OPUF instance $\mathcal{LR}$-$\mathcal{OPUF}^* \leftarrow$ LR-OPUF that is a $(q_\alpha + q_\beta, \varepsilon_1 + \varepsilon_2)$-secure backward and forward unpredictable PUF, and $h(\cdot)$ be an ε_2-secure CRHF. Therefore $(q_\alpha + q_\beta, \varepsilon_1 + \varepsilon_2)$ represents the underlying PUF behaviour. Our protocol $\mathcal{P}$ can assure security against any modelling or machine-learning attacks since underlying PUF behavior is unpredictable.

Proof: To demonstrate this theorem, we leverage the backward-and-forward unpredictability game described earlier. In this game, an adversary $\mathcal{A}$ is granted access to the LR-OPUF connected to the device and acquires a set of CRPs from it. Let $\mathcal{A} = (\mathcal{A}_\alpha, \mathcal{A}_\beta)$ be an adversary with a non-negligible probability of breaking the LR-backward OPUF's and forward unpredictability. We construct an adversary $\mathfrak{B}$ that aims to compromise the underlying physical PUF's unpredictability while achieving collisions in $h(\cdot)$ with the same success probability as

TABLE II
COMPARISON WITH OTHER STATE-OF-THE-ART MODELLING-ATTACK RESILIENT PROTOCOLS BASED ON DESIRABLE FEATURES (DF)

Schemes	DF1	DF2	DF3	DF4	DF5	DF6	DF7
Yu et al. (Protocol #1) [20]	Yes	No	No	Partially	No	No	No
Yu et al. (Protocol #2) [20]	Yes	No	No	Partially	No	No	No
Liang et al. [22]	Yes	No	No	Partially	No	No	No
Gu et al. [23]	Yes	Yes	No	Partially	No	No	No
Zalivaka et al. [40]	Yes	No	No	Partially	No	No	No
Barbareschi et al. [41]	No	Yes	No	No	No	No	No
Wang et al. [42]	Yes	No	No	Partially	No	No	No
Proposed Scheme	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DF1: Mutual Authentication; **DF2:** Resilience Against Replay Attacks; **DF3:** Forward/Backward Unpredictability; **DF4:**Resilience-Against-modelling-Attacks; **DF5:** Privacy-of-the-Device;**DF6:**Scalability; **DF7:**Hides-CRPs-from-the-Verifier;							

$\mathcal{A}$. $\mathcal{A}$ is allowed to query the LR-OPUF up to $(q_\alpha + q_\beta)$ times using challenges C_j $(1 \leq j \leq q_\alpha + q_\beta)$ and receives LR-OPUF responses Q_j. $\mathfrak{B}$ chooses an arbitrary LR-OPUF state ς and provides it to $\mathcal{A}_\alpha$. It then executes a black-box simulation of the backward and forward unpredictability game's challenger $\mathcal{C}$ (as defined in Definition 2). During the simulation, $\mathfrak{B}$ queries the LR-OPUF with a challenge C_j received from $\mathcal{A}_\alpha$, stores (C_j, Q_j) in a log file $\mathcal{F}$, and forwards Q_j to $\mathcal{A}_\alpha$. This process continues until $\mathcal{A}_\alpha$ eventually halts and writes a log file $\mathcal{F}^*$. Afterward, $\mathfrak{B}$ updates the LR-OPUF state to ς^* to change its configuration for subsequent simulations or interactions. When $\mathcal{A}_\beta$ sends a challenge C_j, $\mathfrak{B}$ queries the LR-OPUF, stores (C_j, Q_j) in a log file $\mathcal{F}$, and forwards R_j to $\mathcal{A}_\beta$. Eventually, $\mathcal{A}_\beta$ terminates and outputs a CRP $(C^\#, Q^\#)$ of the LR-OPUF. However, since $\mathfrak{B}$ has never queried $C^\#$ to the LR-OPUF, this contradicts the unpredictability property of the LR-OPUF. Therefore, the success probability of $\mathfrak{B}$ is similar to that of $\mathcal{A}$, which is $\varepsilon_1 + \varepsilon_2$, and $\mathfrak{B}$ makes $(q_\alpha + q_\beta)$ queries to the underlying PUF. The security of the proposed reconfigurable authentication scheme against any ML-attack relies on the unpredictability property of the LR-OPUF. After each round of the authentication process, the LR-OPUF's configuration is updated, ensuring its unpredictability. Due to the input and output transformations, an adversary is unable to predict any PUF response given an input. Therefore, the adversary cannot distinguish the LR-OPUF outputs Q_i/Q_{i+1} and K_i/K_{i+1} from a random string. It is important to note that both (Q_i, Q_{i+1}) and (K_i, K_{i+1}) are independently generated, further enhancing the security of the scheme. Therefore, no adversary can correlate them. Also, the adversary (including the semi-honest verifier/server) can only obtain the transformed output instead of the original PUF response for a given input, and after each reconfigurable operation, the behavior of LR-OPUF changes. Hence, it will be difficult for any polynomial-time adversary $\mathcal{A}$ to model the LR-OPUF.■

V. DISCUSSION

A. Performance Comparison

In this section, we compare the proposed solution with respect to three state-of-the-art protocols presented in [20], [22], [23], [39], [40] and [41], which claim to support resilience against any modelling or ML attacks. We first consider some of the desirable features (DF), such as mutual authentication, resilience against replay attacks, the privacy of the devices, etc., which are imperative for any IoT-based authentication protocol. In [20], the authors have proposed two lockdown authentication protocols, and we refer to them as Protocol #1 and Protocol #2. From Table II, we can see that, like our proposed scheme, the authentication protocols presented [20], [22], [23], [39], and [41] also support the mutual authentication feature. However, Protocol #1 presented in [20] and the protocol presented in [22] are susceptible to replay attacks. More precisely, in Protocol #1 of [20], the device sends its identity (defined as id in [20]) in Step 1 of the protocol to communicate with the server. An adversary $\mathcal{A}$ can intercept that message and resend it later, but the server cannot comprehend that. In addition, when the server replies with the challenge (defined as c in [20]) and half of the response bits (defined as $r1$ in [20]) in Step 2 of Protocol #1, $\mathcal{A}$ can also intercept that message and resend it later. Therefore, the protocol #1 presented in [20] cannot ensure security against replay attacks. However, a similar problem also exists in their Protocol #2. On the other hand, the protocol presented in [22], [39] and [41] uses the same "*seed*"/nonce value to generate the PUF response for each round of authentication, making it vulnerable against replay attacks. Conversely, as discussed in Section IV, if an adversary $\mathcal{A}$ intercepts and replays any of the messages M_1, M_2, and M_3 of our protocol, the recipient will be able to detect that. Next, since the protocol presented in [22] uses the *same* CRP for all sessions, it cannot ensure forward secrecy. Now, Protocol #1 presented in [20] can ensure security against *any learner*. Conversely, Protocol #2 and the protocol presented in [22] ensure security against *heuristic learners*. However, in both the protocols of [20], and the protocols presented in [22], [23], [39], [40] and [41], the semi-honest server is allowed to obtain a considerable number of original CRPs. Then, they can model the PUF behaviour and even form a *marionette PUF* [35], which makes the PUF useless. In [40], when a specified chain of challenges is utilised, the server authenticates the device by XORing the PUF response with the prior responses. While the PUF advantage of not having to store responses is exploited, chains can still be used to model the PUF.

Unfortunately, this makes the protocol vulnerable to device impersonation attacks. In the proposed scheme, we have addressed this issue by transforming (masking) the input/output of the underlying PUF. The proposed scheme is designed based on the underlying foundation of the LR-OPUF, which ensures backward/forward unpredictability. Therefore, the adversary will not be able to predict the behavior of the reconfigured LR-OPUF

TABLE III
COMPARISON WITH OTHER STATE-OF-THE-ART MODELLING-ATTACK RESILIENT PROTOCOLS BASED ON OTHER METRICS

Schemes	PUF-Used	Number of Interaction	Communication Cost
Yu et al. (Protocol #1) [20]	SRAM -PUF	4	512-bit
Yu et al. (Protocol #2) [20]	XOR-PUF	4	640-bit
Liang et al. [22]	TSMACA-PUF	4	3008-bit
Gu et al. [23]	Two Arbiter PUFs	4	768-bit
Zalivaka et al. [40]	Arbiter PUF	4	640-bit
Barbareschi et al. [41]	Any Weak PUF	4	1152-bit
Wang et al. [42]	DCH PUF	4	640-bit
Proposed Scheme	(LR-OPUF (with an underlying strong PUF))	3	1280-bit

TABLE IV
ESTIMATED TOTAL COST OF OUR PROPOSED SCHEME

Setting#	Components	LUTs	DFFs	Power Cons. (W)	Frequency (Mhz)
1	**- LS-PUF :** 64-bit **- Input C:** 64-bit **-** $TF_1(\cdot)$ **and** $TF_2(\cdot)$**:** ASCON (64-bit) **- FE:** BCH Encoding **AES-GCM:**AEAD Encryption 128-bit	2369	3439	0.289	200
2	**- LS-PUF :** 64-bit **- Input C:** 128-bit **-** $TF_1(\cdot)$ **and** $TF_2(\cdot)$**:** ASCON (128-bit) **- FE:** BCH Encoding **AES-OCB:**AEAD Encryption 128-bit	2378	3558	0.386	200
3	**- LS-PUF:** 64-bit **- Input C:** 64-bit **-** $TF_1(\cdot)$ **and** $TF_2(\cdot)$**:** Keccak-f[200] (64-bit) **- FE:** BCH Encoding **AES-EAX:**AEAD Encryption 128-bit	1902	954	0.264	200
4	**- LS-PUF:** 64-bit **- Input C:** 128-bit **-** $TF_1(\cdot)$ **and** $TF_2(\cdot)$**:** Keccak-f[400] (128-bit) **- FE:** BCH Encoding **XChaCha20Poly1305:**AEAD Encryption 128-bit	2245	1355	0.516	200

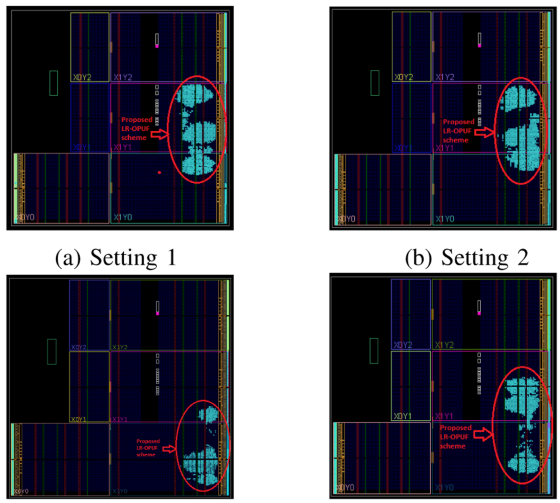
even if she/he knows/obtains a considerable amount of CRPs of the PUF before the reconfiguration (feias shown in Table IV (confused)). Hence, it can ensure security against modelling attacks. Next, from Table II, we can see that none of the protocols presented in [20], [22], [23], [39], [40] and [41] has considered the privacy of the devices. In addition, the protocols presented in [20], [22], [23], [39], [40] and [41] are not scalable. For instance, since Protocol #1 (in [20]) is constructed upon a weak-PUF (SRAM) with a limited number of CRPs, this protocol can only support a limited number of sessions (defined as d in [20]). On the other hand, Protocol #2 in [20] and the protocol presented in [22], [39], are based on the train-PUF-model, where, for authenticating each device, the server needs to load the train-PUF-model specific to that device. For example, instead of storing CRPs, the server in [39], and [41] stores an Arbiter-PUF model for each device. During authentication, the server needs to load the Arbiter-PUF model corresponding to the device, which impacts not only the execution time but also the storage and the enrollment time/process required for each device. In addition, each training process takes a considerable amount of time, which is much higher than any key-sharing or CRP sharing process. This is infeasible in typical IoT environments, where a server may need to authenticate thousands of devices, and the server needs to construct a train-PUF model for each device. Hence, both of these solutions cannot ensure scalability.

Next, we consider some other metrics in order to compare the proposed scheme w.r.t. the protocols presented in [20], [22], [23], [39], [40] and [41]. In this regard, we first consider the PUF used in each design of the authentication protocol. Protocol #1 presented in [20] is constructed using an SRAM-PUF, Protocol #2 is designed based on the XOR-PUF, and the protocol presented in [22] uses a new PUF construction called TSMCA-PUF. On the other hand, the protocol presented in [23] uses two Arbiter PUFs (APUFs), where one of them is used as a genuine PUF and the other as a fake. In the protocol presented in [39], and [41], the server stores an Arbiter-PUF-model for each device for authentication. In the proposed scheme, LR-OPUF is used with an underlying non-reconfigurable strong PUF to validate the legitimacy of the entities (both device and server). From Table III, we can see that the number of interactions required during the execution of the proposed scheme is 3, whereas the other protocols (presented in [20], [22], [23], [39], [40] and [41]) need 4-rounds of interactions. Next, we consider the communication cost of each protocol, which denotes the total size of the messages transmitted during the execution of the protocol. The communication cost of the proposed scheme can

be computed as $\sum_{i=1}^{3} \mathrm{M_i} = 1280$ bits. Table III also shows the communication cost of the protocols (Protocol #1 and Protocol #2 in [20], protocol in [20], and the proposed scheme) during their execution. We note that the protocols presented in [20] have lower communication cost as compared to others. However, it should be noted that, unlike the proposed scheme, the protocols presented in [20], [22], [23], [39], [40] and [41] have not considered any communication security of the messages exchanged during their execution and that makes them vulnerable to many attacks such as replay and forgery attacks. In addition, none of them can ensure the privacy of communication.

B. Experimental Evaluation

The security of the proposed reconfigurable authentication protocol is based on the underlying foundation of the LR-OPUF scheme. Hence, this section presents experimental results to show that LR-OPUF is effective against modelling attacks even if the underlying PUF could be vulnerable to such attacks. To generate these results, we first implemented a 64-bit LS-PUF (Arbiter PUF with 5-XOR outputs) [36] on a Xilinx Zynq-7000 FPGA device using Xilinx Vivado Design Suite. The FPGA is set up on a SASEBO G-II board, where we obtained the responses against the challenges through a RS-232 C cable connected to the SASEBO G-II board. We collected 70,000 CRPs, which were used to train a PUF model. To implement the LR-OPUF, we used another 64-bit LS-PUF implementation on the SASEBO G-II board consisting of a Xilinx Zynq-7000 FPGA hardware platform. The hardware engine also consisted of two 256-bit 2 T MTP NVM (used as logic NVM) for storing the state and the helper data, respectively. The hardware overhead includes the transformation functions ($TF_1(\cdot)$ and $TF_2(\cdot)$), Strong PUF utilized, and FE, where we benchmark our scheme using the hardware overhead of 64-bit Arbiter PUFs. To ensure resource efficiency, the ASCON hashing algorithm and lightweight permutations of the Keccak hash function (Keccak-*f*[200] and Keccak-*f*[400]) are used as transformation functions in the control logic. The (63,30,13)-BCH code [38] is utilized to generate helper data for the fuzzy-extractor construction. For the error correction component of the FE, we employ an LFSR-based implementation of the BCH encoding technique. This encoding function transforms a 30-bit seed's randomness into a 63-bit code word. To ensure a more comprehensive hardware benchmarking, we define different settings (Settings 1 through 4) to encompass a wide range of scenarios. In Table IV, we provide a total of resource requirements for all settings in Look-Up Tables (LUTs) and D-Flip-Flops (DFFs) for implementing the proposed LR-OPUF construction. In all Settings, the PUF and the error correction part of FE are identical; however, we employ distinct **input C** and transformation function variations (Keccak and ASCON hash functions). Note that because of the algorithm design structures, the Keccak hash functions exhibit slightly higher power consumption than the ASCON hashing algorithm, but they fall within acceptable limits. For AEAD encryptions, we have used four different modes: AES-GCM, AES-OCB, AES-EAX, and XChaCha20Poly1305. Among all of them, XChaCha20Poly1305 has a much greater limit on the number of messages and message size.



(a) Setting 1 (b) Setting 2

(c) Setting 3 (d) Setting 4

Fig. 4. Layouts of the Proposed LR-OPUF Scheme (Settings 1 through 4) on a Xilinx Zynq-7000 FPGA Device

Based on the information provided in Table IV, it is evident that for a 64-bit **input C**, Setting 3 has the lowest hardware requirement. Moreover, when it comes to a 128-bit **input C** utilizing the Keccak-*f*[400] as the transformation function, Setting 4 is regarded as an acceptable choice in relation to hardware requirements compared to Setting 2. Fig. 3 indicates the chip layouts for each setting in the synthesized LR-OPUF scheme. The chip layouts corresponding to Setting 1, Setting 2, Setting 3, and Setting 4 are represented by Fig. 4(a), (b), (c), and (d), respectively.

Now, after obtaining the PUF responses from the underlying Arbiter-PUF (LS-PUF) of the LR-OPUF, we apply the BCH code for error correction, followed by hashing to construct the final output. Both the Arbiter-PUF (LS-PUF) and the LR-OPUF are subjected to the same set of challenges, consisting of 70,000 Challenge-Response Pairs (CRPs). To model the behaviours of both PUFs, we employ the Scikit-learn machine-learning library in Python, utilizing the Jupyter Notebook as our Integrated Development Environment (IDE). We use four well-known supervised learning algorithms: Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM) and Evolution Strategy (ES). The objective of this evaluation is to predict the correct response for a given challenge with high accuracy using these machine-learning algorithms. For training and testing, we split the generated CRP set of 70,000 samples into two parts: the training set, which comprises 80% of the CRPs, and the test set, containing the remaining 20% . Table V demonstrates that the accuracy of the four methods is relatively high for LS-PUF, indicating their vulnerability to modelling attacks. However, in the case of LR-OPUF, despite using the same LS-PUF as the underlying module, the accuracy of these methods is significantly low. This is attributed to the masking of PUF input/output

TABLE V
PREDICTION ACCURACY RESULT (IN %)

Machine Learning Algorithms	LS-PUF [37]	LR-OPUF
Naive Bayes	96.37 %	15.83 %
Logistic Regression	97.62 %	17.62 %
Support Vector Machine	98.72 %	19.89 %
Evolution strategy	98.3 %	24.75%

through the application of two transformation functions (TF_1 and TF_2). Consequently, modelling a reconfigurable PUF like LR-OPUF presents a considerable challenge, making it secure against modelling attacks.

VI. CONCLUSION

Traditionally, PUFs exhibit static change-response behaviour. However, many use cases/applications require dynamic PUF challenge-response behaviour that can be achieved from reconfigurable PUFs. In this paper, we first introduced a new reconfigurable PUF construction, which we call LR-OPUF, which is then used for designing a privacy-aware reconfigurable authentication scheme. Through comprehensive security analyses, it is shown that the proposed authentication scheme can ensure several notable properties such as backward/forward unpredictability, authentication without revealing CRPs to the verifier, resilience against man-in-the-middle and modelling attacks, etc. We argue that the proposed scheme can be useful for many applications, such as recyclable/reusable access tokens, PUF-based hardware intrinsic key storage, etc., which can be updated in a secure manner.

REFERENCES

[1] 2015. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] P. Newman, "The Internet of Things," 2020. Accessed: Apr. 04 2020. [Online]. Available: https://www.businessinsider.com/internet-ofthings-report?IR=T

[3] A. Spadafora, "IoT devices still major target for cyberattacks," 2011. Accessed: Apr. 05, 2010. [Online]. Available: https://www.techradar.com/sg/news/iotdevices-still-major-target-for-cyberattacks

[4] P. S. Ravikanth, "Physical one-way functions," PhD dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2001.

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

[6] U. Ruhrmair et al., "Modelling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 237–249.

[7] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 586–591.

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

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

[10] J. Bringer, H. Chabanne, and T. Icart, "Improved privacy of the tree-based hash protocols using physically unclonable function," in *Proc. 6th Int. Conf. Secur. Cryptogr. Netw.*, Berlin, Heidelberg, 2008, pp. 77–91.

[11] P. Tuyls and B. Skoric, "Strong authentication with physical unclonable function," in *Security Privacy and Trust in Modern Data Management*. Berlin, Germany: Springer, 2008, pp. 133–148.

[12] S. Kardas, M. S. Kiraz, M. A. Bingl, and H. Demirci, "A novel RFID distance bounding protocol based on physically unclonable functions," in *Proc. 7th Int. Conf. RFID Secur. Privacy*, 2012, pp. 78–93.

[13] B. Gassend et al., "Controlled physical random functions," in *Proc. Comput. Secur. Appl. Conf.*, 2002, pp. 149–160.

[14] N. Wisiol, "Breaking the lightweight secure PUF: Understanding the relation of input transformations and machine learning resistance," 2019. [Online]. Available: https://eprint.iacr.org/2019/799

[15] N. Wisiol et al., "Splitting the interpose PUF: A novel modelling attack strategy," *IACR Trans. Cryptographic Hardware Embedded Syst.*, vol. 3, pp. 97–120, 2020.

[16] 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 Secur.*, vol. 13, no. 11, pp. 2831–2843, Nov. 2018.

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

[18] D. Moriyama, S. Matsuo, and M. Yung, "PUF-based RFID authentication secure and private under complete memory leakage," *IACR Cryptol. ePrint Arch.*, vol. 2013, 2013, Art. no. 712, [Online]. Available: http://eprint.iacr.org/2013/712

[19] A. Aysu, E. Gulca, D. Moriyama, P. Schaumont, and M. Yung, "End-to-end design of a PUF-based privacy preserving authentication protocol," *Cryptographic Hardware Embedded Syst.*, vol. 9293, pp. 555–576, 2015.

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

[21] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "A lightweight, robust, secure authentication by substring matching," in *Proc IEEE Symp. Secur. Privacy Workshops*, 2012, pp. 33–44.

[22] W. Liang et al., "A double PUF-based RFID identity authentication protocol in service-centric Internet of Things environments," *Inf. Sci.*, vol. 503, pp. 129–147, 2019.

[23] C. Gu et al., "A modelling attack resistant deception technique for securing PUF based authentication," in *Proc. Asian Hardware Oriented Secur. Trust Symp.*, 2019, pp. 1–6.

[24] K. Kursawe, "Reconfigurable physical unclonable functions - enabling technology for tamper-resistant storage," in *Proc. IEEE Int. Workshop Hardware-Oriented Secur. Trust*, 2009, pp. 22–29.

[25] U. Ruhrmair and D. E. Holcomb, "PUFs at a glance," in *Proc. Des. Automat. Test Europe Conf. Exhib.*, 2014, pp. 1–6.

[26] G. Becker, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *Proc. 17th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2015, pp. 535–555.

[27] M. Roel, *Physically Unclonable Functions: Constructions, Properties and Applications*, Berlin, Germany: Springer, 2013.

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

[29] F. Ganji, S. Tajik, and J. P. Seifert, "Why attackers win: On the learn-ability of XOR arbiter PUFs," in *Proc. Int. Conf. Trust Trustworthy Comput.*, 2015, pp. 22–39.

[30] B. Blanchet, "Modelling and verifying security protocols with the applied pi calculus and ProVerif," *Foundations Trends R. Privacy Secur.*, vol. 1, pp. 1–2, 2016, doi: 10.1561/3300000004.

[31] C. Brzuska et al., "Physical uncloneable functions in the universal composition framework," in *Proc. Annu. Cryptology Conf.*, 2011, pp. 51–70.

[32] D. C. Ranasinghe et al., "Security and privacy: Modest proposals for low-cost RFID systems," in *Proc. Auto-ID Labs Res. Workshop*, 2004, pp. 221–226.

[33] B. Skoric, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," *Appl. Cryptogr. Netw.*, vol. 3531, pp. 407–422, 2005.

[34] J. Miskelly and M. O'Neill, "Fast DRAM PUFs on commodity devices," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 39, no. 1, pp. 3566–3576, pp. 3566–3576, Nov. 2020, doi: 10.1109/TCAD.2020.3012218.

[35] U. Ruhrmair and M. Dijk, "PUFs in security protocols: Attacks Models and security evaluations," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 286–300.

[36] M. Majzoobi et al., "Lightweight secure PUFs," in *Proc. 2008 IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2008, pp. 670–673.

[37] S. Sutar et al., "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication and random number generation," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 1, 2017, Art. no. 17.

[38] J. Delvaux, D. Gu, I. Verbauwhede, and M. Hiller, "Efficient Fuzzy Extraction of PUF-Induced Secrets: Theory and Applications," in *Cryptographic Hardware and Embedded Systems (CHES)*. Berlin, Germany: Springer, 2016, pp. 412–430.

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

[40] M. Barbareschi et al., "A PUF-based mutual authentication scheme for cloud-edges iot systems," *Future Gener. Comput. Syst.*, vol. 101, pp. 246–261, 2019.

[41] Y. Wang et al., "A dynamically configurable PUF and dynamic matching authentication protocol," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1091–1104, Second Quarter 2022, doi: 10.1109/TETC.2021.3072421.