A Comparative Study of Design Paradigms for PUF-based Security Protocols for IoT Devices: Current Progress, Challenges and Future Expectation

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

***Abstract*—Device authentication is an essential security feature for the Internet of Things (IoT). Many IoT devices are deployed in the open and in public places, which makes them vulnerable to physical and cloning attacks. Physical Unclonable Functions (PUFs) have emerged as a promising technology to address the challenges in the development of lightweight authentication protocols in these environments. PUFs can facilitate high levels of security, while simultaneously minimizing the computational resource requirement per device. This article presents the current progress and challenges in designing PUF-based authentication protocols for IoT-devices. We first present state-of-the-art design approaches for constructing secure authentication protocols by considering ideal-PUFs, noisy-PUFs, and Machine Learning attacks on PUF. Subsequently, we describe the challenges faced by these approaches and future expectations for designing PUF-based security solutions.**

***Index Terms*—Mutual authentication, Physically uncloneable functions, IoT.**

I. Introduction

Consumers and industries have benefited significantly from a diverse range of autonomous electronic devices. Many of these devices are interconnected by communication technologies, thereby forming the Internet of Things (IoT) [1]. The IoT has started to become pervasive in our daily lives, finding applications in industries, homes, medicine, public infrastructure, etc. With the increasing deployment of IoT devices and systems, security issues associated with them have gained prominence, and security incidents that exploit IoT devices have drawn increasing public attention. Security for IoT devices is challenging due to two main factors: (i) Many IoT devices have limited processing, storage, and energy capabilities, which limit the applicability of existing cryptographic tools; (ii) IoT devices generally operate without any human intervention and thus have to store secret keys in their memory, making them vulnerable to a number of physical and side channel attacks. For example, keys stored in the memory may be read off a physically captured device and used by an adversary to launch attacks. Therefore, security against physical attacks is a major concern in IoT.

P. Gope, is with Department of Computer Science, National University of Singapore, 21 Lower Kent Ridge Rd, Singapore 119077. (E-mail: prosana.nitdgp@gmail.com)

B. Sikdar is with Department of Electrical and Computer Engineering, National University of Singapore, 21 Lower Kent Ridge Rd, Singapore 119077. (Email: bsikdar@nus.edu.sg)

A. PUF as an Imperative Security Primitives for IoT

The use of hardware security primitives for generating and processing secret keys is a promising way to provide security to IoT devices in face of the complex threat landscape that they face. Since they were first proposed in 2002, physical unclonable functions (PUFs) [2-3] have emerged as a leading contender for hardware-based security solutions. The operation of PUFs is based on exploiting the variations in the circuit-level micro-structures that are created during the integrated circuit (IC) manufacturing process to provide functions that uniquely (at the chip level) map an input binary string to an output. It is impossible to control these micro-structural variations in the IC during the manufacturing process (even by using the same equipment and photo-lithography mask), thereby providing the basis of security for PUFs. As the PUF output depends on the physical characteristics of the IC, any attempt to tamper with the PUF changes its behavior and renders the PUF useless. Due to this unique property, PUFs have gained popularity as a paradigm for physical security of resource constrained IoT devices such as RFID tags, sensors, etc.

PUF-based authentication protocols for IoT systems have to consider the following requirements: (i) avoid the storing of secret keys in the device memory (where the secret is required during the execution of the protocol); (ii) ensure security even under noisy conditions; (iii) resilience against machine-leaning or modeling attacks; (iv) scalability so that it can work for IoT applications with a large number of devices. Even though PUFs have emerged as a promising security primitive for ensuring physical security of IoT devices (e.g., against tampering and cloning), this article highlights several challenges that PUF-based authentication protocol may face. To provide context, we first present four state-of-art design approaches for constructing authentication protocols for IoT devices by considering ideal-PUFs, noisy-PUFs, and machine learning attacks on PUFs. Subsequently, we describe possible challenges that security protocols based on these approaches may encounter. Finally, this article describes future expectations for designing PUF-based security solutions. The contributions of this article can be summarized as:

- It presents state-of-art methodologies for using PUFs in authentication protocols for IoT devices with physical security issues while considering challenges such as noise

and machine learning attacks.
- It presents a comparative analysis of the design approaches.
- It outlines open research issues and future expectations for PUF-based security solutions for resource constrained IoT devices.

II. Preliminaries

A. Physical Unclonable Functions

PUFs exploit the intrinsic random variability in the physical micro-structure of ICs to produce a unique output in the form of a "response", R, to an input called the "challenge", C. PUFs can be considered as a challenge-response system modeled as $R = f(C)$ where the function $f(\cdot)$ models the input-output relation of the PUF. The function $f(\cdot)$ is governed by the variations in the circuits' internal parameters. PUF security leverages on the difficulty of measuring or estimating these parameters and the difficulty of creating two chips with the same parameters (i.e., unclonability). A PUF may be classified as either strong or weak, based on the number of unique challenges it can process. A weak PUF can only process a small number of challenges while a strong PUF has the ability to process a number of challenges that is large enough to make the complete measurement of all challenge-response pairs (CRPs) within a limited time-frame unfeasible.

One of the first PUF implementations was an optical PUF where the response (speckle pattern) depends on the input laser location/polarization (challenge) [2]. Owing to their higher cost and the need for well-calibrated external measurement devices, subsequent efforts have focused on the development of semiconductor based PUFs. Examples of silicon based PUFs that exploit manufacturing variability in gate and wire delays as the source of unclonable randomness include arbiter PUFs and its variations (e.g., XOR arbiter PUFs) and ring oscillator PUFs. Other examples of PUFs include mismatch based silicon PUFs such as SRAM PUFs, latch PUFs, ip-op PUFs, butterfly PUFs, and analog PUFs based on silicon such as current-based PUFs and nonlinear current mirror based PUFs (that exploit nonlinear characteristics of current or voltage) [4]. Hardware embedded with PUFs as well as PUF-based security solutions are also becoming commercially available (e.g., Xilinx Zynq Ultrascale+ MPSoC, Intrinsic ID, and PUF solutions and designs from Quantum Trace, ICTK, and Quantum Base [new citation, should be [4]]).

Advantages of PUFs over standard secure digital storage include [3]:

- The hardware for PUFs uses simple digital circuits that require lower power and chip-area than EEPROM/RAM solutions with anti-tamper circuitry.
- Security applications using PUFs do not need expensive cryptographic hardware such as those required for efficient execution of secure hash algorithm (SHA) or public/private key encryption algorithms.
- Security of a PUF is derived from the physical microstructure of the chip, making the execution of invasive attacks more difficult.
- Producing a physical clone of a PUF is extremely hard.
- PUFs are easier to manufacture and operate than non-volatile EEPROMs or battery-backed RAMs that require external always-on power sources.

B. Fuzzy Extractor

One of the limitations of PUFs is their sensitivity to ambient and operating conditions such as temperature and voltage levels. As a result, the response of a PUF to the same challenge may vary based on prevailing conditions. Fuzzy extractors have been proposed as an efficient solution for this problem. A fuzzy extractor [10-12] converts a PUF's output into a uniform pseudorandom secret that serves as a secret key during the authentication phase. Pseudorandom functions are applied in our protocol for creating uniformly random session keys. A fuzzy extractor (d, λ, ϵ) consists of two procedures: Gen$(\cdot)$ and Rec$(\cdot)$. Gen$(\cdot)$ outputs a keying element K and helper data hd, i.e., $(K, hd) = \mathrm{FE.Gen}(R)$ for a given input bit string R. Rec$(\cdot)$ takes a noisy input R' and helper data hd and outputs the key K, i.e., $K = \mathrm{FE.Rec}(R', hd)$, if the Hamming distance between R' and R is at most d.

C. Reverse Fuzzy Extractor

The underlying error decoding algorithms of fuzzy extractors and secure sketches are typically complex and time consuming. Thus, reverse fuzzy extractors have been developed for fast implementation of secure sketch and fuzzy extractors [7]. With reverse fuzzy extractors, PUF-enabled devices need not perform the computationally intensive reconstruction algorithm. Instead, the devices execute the helper data generation algorithm. Consequently, a new helper data hd is generated each time a PUF is queried and the verifier corrects the reference value R of its database to the noisy PUF response R', which is different each time the PUF is evaluated.

D. Fractional Hamming Distance

Hamming weight, $\mathrm{HW}(R)$, counts the number of 1s in vector R. Let $L(R)$ denote the length of R, as used in, e.g., the fractional Hamming distance $\mathrm{FHD}(R, R^*) = \mathrm{HW}(R \oplus R^*)/L(R)$ between vectors R and R^*.

E. Pseudorandom Function

A pseudorandom function PRF: $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^{k'}$ takes a secret key $sk \in \{0,1\}^k$ and a message $m \in \{0,1\}^*$ and provides an arbitrary string $\mathrm{PRF}(sk, m)$ that is indistinguishable from a random string. Security of the PRF can be defined by the following game between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. Challenger $\mathcal{C}$ first selects a coin $b \xleftarrow{U} \{0,1\}$ and secret key $sk \xleftarrow{U} \{0,1\}^k$. Then, $\mathcal{C}$ creates a truly random function RF. The adversary $\mathcal{A}$ can adaptively issue an oracle query to the challenger $\mathcal{C}$ to obtain a response from a function. When $\mathcal{A}$ sends m, then $\mathcal{C}$ responds with $\mathrm{PRF}(sk, m)$ if $b = 1$. On the other hand, if $b = 0$, then $\mathcal{C}$ inputs m to RF and responds with its output. Finally, $\mathcal{A}$ outputs a guess b' and wins the game if $b' = b$. Here, the advantage of the adversary to win the game can be defined as $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PRF}}(k) = |2\Pr[b' = b] - 1|$. A PRF is said to be $\epsilon-$secure if for any polynomial time adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PRF}}(k) \leq \epsilon$ holds.

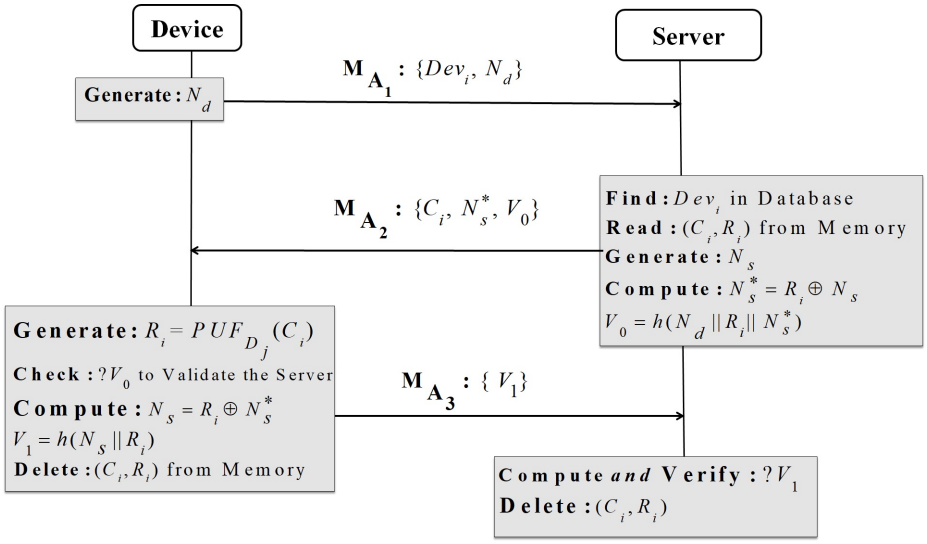

Figure 1. Authentication under ideal PUF conditions (Protocol 1).

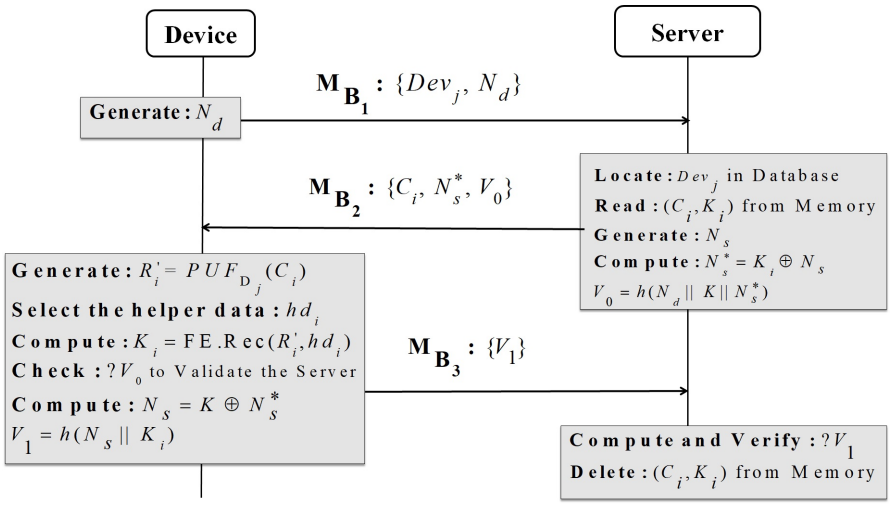

Figure 2. Authentication under noisy PUF condition with fuzzy extractor (Protocol 2)

III. Construction of PUF-based Authentication Protocols for IoT devices

Depending upon the constructional requirements, PUF-based authentication protocols can be divided into four categories: ideal PUF-based construction, noisy PUF-based construction with fuzzy extractor, reverse fuzzy extractor-based construction, and the machine-learning resilience-based construction. The ideal PUF-based design protocol provides a baseline for the design of PUF-based protocols and is useful for supporting the authentication process by generating secret stable PUF-response at run-time under controlled operating conditions (discussed in detail in Section III-F). In cases where the operating conditions are variable and/or noisy, fuzzy extractor and reverse fuzzy extractor-based protocols can be used for authentication. However, if the execution of the FE.Rec algorithm is too heavy for an IoT device and the device can afford a non-volatile-memory (NVM) for storing the secret key, then the reverse fuzzy extractor-based protocol is more preferable than fuzzy extractor and can also ensure two-factor security. Finally, protocols based on the above three approaches are insecure against machine-leaning (ML) or modeling attacks (discussed in detail in Section III-D). Therefore, we need a ML-attack-resilience-based authentication protocol. In this section, we first present these four approaches for designing PUF-based authentication schemes for IoT devices. Note that most of the existing PUF-based protocols (such as [4-8]) use these approaches in their design construction. Subsequently, we highlight the challenges faced by each of these approaches.

A. Protocol 1: Ideal PUF-based Construction

This section presents the design methodology for a lightweight authentication scheme for IoT devices when the PUF is ideal [4-5].

Setup Phase: Execution of this phase is carried out through a secure channel. To start execution of this phase, an IoT device D_j sends it's identity to the server and requests the server for registration. After receiving the request, the server

generates a set of challenges $C = \{C_1, C_2, \cdots, C_n\}$ and then sends C to the device. Upon receiving the request, the device extract the PUF outputs $R = \{R_1, R_2, \cdots, R_n\} = PUF_{D_j}(C_1, C_2, \cdots, C_n)$, and sends R to the server. Then, the server stores all the challenge-response pairs (CRPs) $(C, R) = \{(C_1, R_1), (C_2, R_2), \cdots, (C_n, R_n)\}$ for n interactions with the device. Note that when all the CRPs are used up, then the server generates a new set of challenges and asks the device to send the corresponding responses through the execution of this phase.

Authentication Phase: The device generates a nonce N_d and then sends N_d along with its id D_j to the server. After receiving the request, the server first searches its database for D_j and its security credentials, and then selects a CRP (C_i, R_i). The server then generates a nonce N_s and computes $N_s^* = R_i \oplus N_s$ and $V_0 = h(N_d||R_i||N_s^*)$. Finally, the server composes a message $\mathrm{M_{A_2}}$: $\{C_i, N_s^*, V_0\}$ and sends it to the device. Upon receiving $\mathrm{M_{A_2}}$, the device first uses the challenge C_i and generates its PUF response $R_i = PUF_{D_j}(C_i)$. It then computes and verifies the parameter V_0. If the verification is unsuccessful, then the device aborts the execution of this phase. Otherwise, the device computes $N_s = R_i \oplus N_s^*$ and $V_1 = h(N_s||R_i)$, and sends V_1 to the server. Upon receiving V_1, the server validates it. If the validation is successful, the server authenticates the device. Then, both the device and the server delete the CRP (C_i, R_i). Details of this phase are shown in Fig. 1.

Challenges in this construction: First consider the case where the PUF (PUF_{D_j}) used in this construction is a weak-PUF (such as SRAM). In this scenario, the security of the system is compromised when the responses arising from the PUF are read out by invasive means. This is in principle comparable to the security of a secret key stored in NVM, even though the PUF-response exists in the system only for a short time. This inherent attack point of weak PUFs has been successfully exploited in literature (e.g., [17]). Even if care is taken to prevent SRAM PUF values from ever being read over standard on-chip channels, attacks using laser stimulation can reveal cell states in a powered SRAM PUF [17].

In the case where the PUF (PUF_{D_j}) used in this construction is a strong-PUF, the protocol is susceptible against machine-learning or modeling attacks (as discussed in Section III.D). Note that the cloning and invasive attacks that are practical on weak PUFs, are less applicable on strong PUFs since the adversary requires knowledge of all possible CRPs (a large number) in order to perform such attacks (and weak PUFs are limited in the number of CRPs). Finally, although differential design methodologies do improve reliability, noise is still a factor in PUF design. Even in optimal environmental conditions, noise will result in one or several of the output bits of the PUF to be incorrect for any given challenge. Hence, the scheme above will not work for noisy PUF conditions. Modern PUF designs employ multiple error-correction techniques to correct bits for improving reliability. The following sections present two approaches for designing authentication schemes for noisy PUF conditions.

B. Protocol 2: Noisy PUF-based Construction With Fuzzy Extractor

This section presents a construction for lightweight authentication schemes for IoT devices using fuzzy extractor under noisy PUF conditions [5-6].

Setup Phase: To start execution of this phase, an IoT device D_j sends it's identity to the server and requests for registration through a secure channel. Upon receiving the request, the server generates a set of random challenges $C = \{C_1, C_2, \cdots, C_n\}$ and sends C to the device. Upon receiving C, the device extracts the PUF outputs $R = \{R_1, R_2, \cdots, R_n\} = PUF_{D_j}(C_1, C_2, \cdots, C_n)$, and sends R to the server through a secure channel. Next, for each CRP (C_i, R_i), the server computes $(K_i, hd_i) = \mathrm{FE.Gen}(R_i)$ and composes the set of challenge-helper data pairs $(C, hd) = \{(C_1, hd_1), (C_2, hd_2), \cdots, (C_n, hd_n)\}$ and the challenge-key data pairs $(C, K) = \{(C_1, K_1), (C_2, K_2), \cdots, (C_n, K_n)\}$, and sends the challenge-helper data pairs (C, hd) to the device. Finally, the server stores $\{D_j, (C, K)\}$ in its database and the device needs to store (C, hd) in its memory for further interactions.

Authentication Phase: The IoT device generates a random number N_d and then sends N_d along with its identity D_j to the server. Upon receiving the request, the server first finds the security credentials of device D_j and then selects a challenge-keying data pair (C_i, K_i). After that, the server generates a nonce N_s and computes $N_s^* = K_i \oplus N_s$ and $V_0 = h(N_d||K_i||N_s^*)$. Finally, the server composes a message $\mathrm{M_{B_2}}$: $\{C_i, N_s^*, V_0\}$ and sends it to the device. Upon receiving message $\mathrm{M_{B_2}}$, the device first uses the challenge C_i and locates challenge-helper data pair (C_i, hd_i). Then, the device generates its PUF response $R_i^{'} = PUF_{D_j}(C_i)$, calculates $K_i = \mathrm{FE.Rec}(R^{'}{}_i, hd_i)$, and then checks the parameter V_0. If the verification is successful, then the device authenticates the server, computes $N_s = K_i \oplus N_s^*$ and $V_1 = h(N_s||K_i)$, and sends V_1 to the server. Upon receiving parameter V_1, the server validates it. If the validation is successful, the server authenticates the device. Finally, the device deletes the challenge-helper data pair (C_i, hd_i) from its memory and the server deletes the challenge-keying data pair (C_i, K_i) from its database. Details of this phase are shown in Fig. 2.

Challenges in this construction: Even though the protocol above works for noisy PUFs, the device needs to execute the computationally expensive reconstruction phase (which limits its applicability for resource limited IoT devices). In addition, in this protocol, the device needs to store the set of challenge-helper data pairs (C, hd) which results in additional storage cost at the device. Also, the fuzzy extractor used by the PUF in this protocol to deal with noise has certain limitations as reported in [16] and this creates some security issues as well. For instance, the security requirement for fuzzy extractors is that the key is uniform even to a (computationally unbounded) adversary who has observed the helper data hd_i. However, this requirement is harder to satisfy as the allowed error tolerance increases, because it becomes easier for the adversary to guess key K_i. This attack is enabled by the functionality of the fuzzy extractor. Therefore, it is important that the fuzzy extractors

should ensure that helper data does not leak the full PUF response.

C. Protocol 3: Noisy PUF-based Construction With Reverse Fuzzy Extractor (Two-Factor Security)

This section shows how reverse fuzzy extractors may be used to construct a lightweight authentication scheme for IoT devices under noisy PUF conditions [7-8].

Setup Phase: Execution of this phase is carried out through a secure channel. First, an IoT device D_j sends its identity to the server. Then, the server generates a set of random challenges $C = \{C_1, C_2, \cdots, C_n\}$, and sends C to the device. Upon receiving the request, the device first extracts the PUF outputs $R = \{R_1, R_2, \cdots, R_n\} = PUF_{D_j}(C_1, C_2, \cdots, C_n)$, and then sends R to the server. Next, the server randomly generates a key, K_{ds}, and sends it to the device. The server stores the key along with the challenge-response pairs (C, R) for the next n interactions with the device.

Authentication Phase: The device generates a random number N_d, computes $N_d^* = K_{ds} \oplus N_d$, composes a request message $\mathrm{M}_{\mathrm{C}_1}$: $\{D_j, N_d^*\}$, and sends $\mathrm{M}_{\mathrm{C}_1}$ to the server. After receiving $\mathrm{M}_{\mathrm{C}_1}$, the server locates the security credentials of D_j and selects the secret key K_{ds} and a challenge-response pair (C_i, R_i). Then, the server generates a nonce N_s, and computes $N_d = K_{ds} \oplus N_d^*$, $N_s^* = K_{ds} \oplus N_s$, and $V_0 = h(N_d||RK_i||N_s^*)$. The server the composes a response message $\mathrm{M}_{\mathrm{C}_2}$: $\{C_i, N_s^*, V_0\}$ and sends it to the device. Upon receiving $\mathrm{M}_{\mathrm{C}_2}$, the device first uses the challenge C_i and computes and verifies the parameter V_0. If the verification is successful, then the device extracts the PUF response $R_i^{'} = PUF_{D_j}(C_i)$ and calculates $N_s = K_{ds} \oplus N_s^*$, $(k_i, hd_i) = \mathrm{FE.Gen}(R_i^{'})$, $hd^* = h(K_{ds}||N_s) \oplus hd_i$, and $V_1 = h(N_s||k_i||hd^*)$. Finally, the device composes a message $\mathrm{M}_{\mathrm{C}_3}$: $\{V_1, hd^*\}$ and sends it to the server. Upon receiving $\mathrm{M}_{\mathrm{C}_3}$, the server computes $hd_i = h(K_{ds}||N_s) \oplus hd_i^*$ and $k_i = \mathrm{FE.Rec}(R_i, hd_i)$, and verifies V_1. If the verification is successful, the server authenticates the device. Lastly, the server deletes the CRP (C_i, R_i) from its database. Details of this phase are depicted in Fig. 3.

Challenges in this construction: The above approach also works for noisy PUFs and each IoT device maintains two factors (secret key K_{ds} and its PUF PUF_{D_j}) for proving its legitimacy to the server. However, in this solution, if the attacker can obtain the secret key K_{ds} stored in the non-volatile memory (NVM) of the device, then the attacker can decrypt the helper data. Repeated exposure of the helper data may result in additional min-entropy loss [12], since the helper data reveals information about the PUF response. Each execution of the helper data generator $\mathrm{Gen}(\cdot)$ on a different noisy version of the same PUF response reveals new helper data. However, reverse fuzzy extractors give no guarantee about the min-entropy of the PUF response in case multiple helper data for different noisy variants of the same response is known [12], [16]. Hence, reverse fuzzy extractors may leak the full PUF response, when $\mathrm{Gen}(\cdot)$ and $\mathrm{Rec}(\cdot)$ are based on a conventional fuzzy extractor. One option to address this issue is to replace the conventional fuzzy extractor with the fractional Hamming distance, which can even lower the computational overhead at the device end (as shown in Table II.)

D. Protocol 4: Train-PUF-Model-based Construction

Machine learning (ML) modeling attacks have emerged as the primary security issue for PUF-based security protocols. In such attacks, the ML algorithm collects and analyzes the CRPs from a PUF and uses them to create a model for the PUF. The attack starts with the adversary generating multiple PUF models with different parameters. These models are then trained using the collected CRPs, and the PUF model with the challenge-response behavior closest to observed CRPs is selected among the generated PUF models. Additionally, the parameters of the selected PUF model are randomly mutated so that new PUF models are generated, and this process is repeated until the final PUF model produces a response similar to that of the original PUF. All three protocols described earlier are vulnerable to such ML-attacks [13-14]. This section presents a train-PUF-model-based [15] authentication protocol that is resistant to ML-attacks (by limiting the ability of attackers to collect the CRPs). In this approach, the PUF supports a mode of operation available only during the setup phase, where, instead of storing explicit CRPs for each device, a train-PUF-model is stored instead. The train-PUF-model extracts a linear amount of manufacturing variation information about the PUF to train an authentication verification model.

Setup Phase: During the setup phase, instead of storing the security credentials for each device D_j, the server stores a train-PUF-model TPM_j. The setup interface is then disabled.

Authentication Phase: During the execution of the authentication phase, the device generates a challenge C_j^d and composes a message $\mathrm{M}_{\mathrm{D}_1}$: $\{D_j, C_j^d\}$ and sends $\mathrm{M}_{\mathrm{D}_1}$ to the sever. Upon receiving $\mathrm{M}_{\mathrm{D}_1}$, the sever uses TPM_j to compute $TPM_j(C_j^d) = R_{j1}^d||R_{j2}^d$, where R_{j1}^d and R_{j2}^d constitute the PUF-response R_j^d (i.e., $R_j^d = R_{j1}^d||R_{j2}^d$). Then, the server generates a new challenge C_j^s, composes a message $\mathrm{M}_{\mathrm{D}_2}$: $\{R_{j1}^d, C_j^s\}$, and sends $\mathrm{M}_{\mathrm{D}_2}$ to the device. Upon receiving $\mathrm{M}_{\mathrm{D}_2}$, the device first generates the PUF response $PUF_{D_j}(C_j^d) = R_{j1}^{d*}||R_{j2}^{d*}$ and if $\mathrm{FHD}(R_{j_1}^{d*}, R_{j1}^d) > \tau$, then the device aborts the execution of the protocol. Otherwise, the device computes $PUF_{D_j}(C_j^s) = R_{j1}^{s*}||R_{j2}^{s*}$, composes a message $\mathrm{M}_{\mathrm{D}_3}$: $\{R_{j2}^{s*}\}$, and sends $\mathrm{M}_{\mathrm{D}_3}$ to the server. The server then uses the train-PUF-model $TPM_j(C_j^d)$ and if $\mathrm{FHD}(R_{j_2}^{s*}, R_{j2}^s) > \tau$, it aborts the authentication process. Otherwise, the server considers the device as legitimate. In this protocol, the server and the device jointly generate the challenge for the PUF, and the attacker cannot impersonate either party for the purpose of completely controlling the PUF challenge information. Details of this phase are depicted in Fig. 4.

Challenges in this construction: Even though the protocol above is secure against ML-attacks, one of its major issues is that of scalability. The server needs to maintain a train-PUF-model for each device, which incurs significant storage cost. Moreover, during the authentication process, the server needs to load the train-model into its memory, which is a time consuming process [15]. Another issue with this protocol

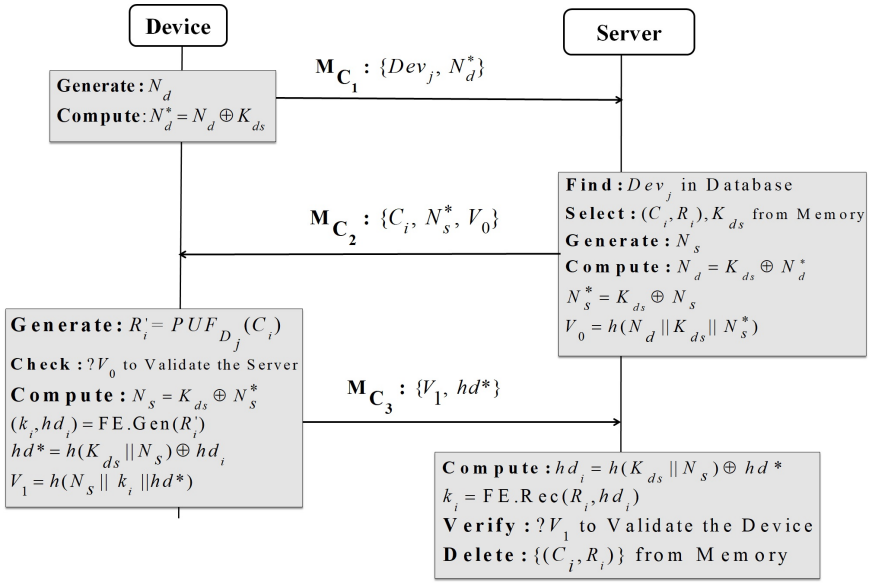

Figure 3. Authentication under noisy PUFs with reverse fuzzy extractor (Protocol 3).

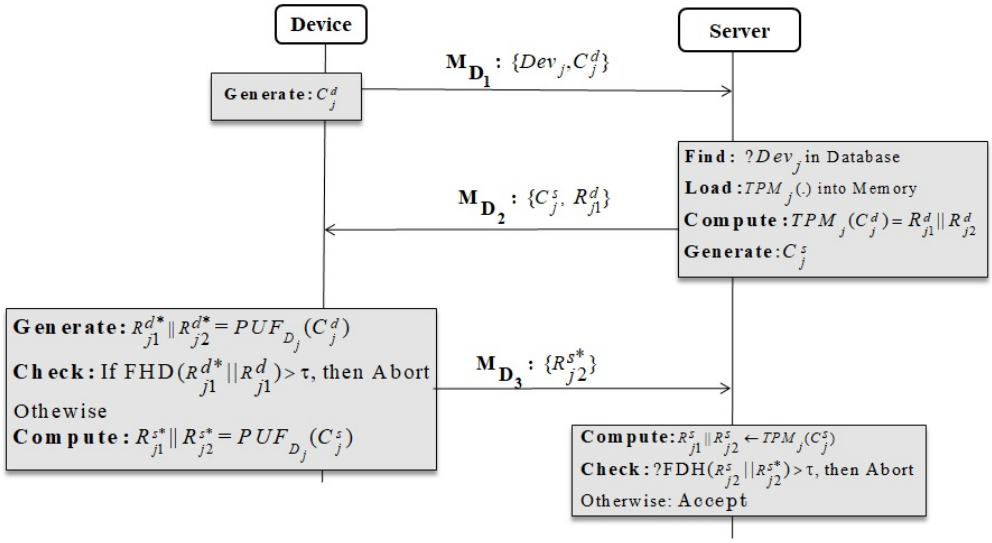

Figure 4. Train-PUF-Model-based authentication protocol for resilience against ML-attacks (Protocol 4).

is the possibility of repeated measurements using the same challenge. This problem can easily be addressed if the server keeps track of the used challenge. However, this requires additional resources since the server needs to record the challenges used in each session of the protocol and both the server and the device are not permitted to reuse the same challenge.

E. Comparative Analysis

This section presents a comparative analysis of the four design approaches of PUF-based protocols for IoT devices. We first compare the design approaches in terms of desirable imperative features (DIF) for IoT-based authentication schemes using PUFs. Table 1 shows that all four protocols ensure mutual authentication. While Protocol 3 achieves most of the DIFs, it does not provide protection against ML-attacks on PUFs. Besides, in this construction the device needs to have a secure NVD for maintaining the secret-key, which would require additional cost at the device end. On the other hand, while Protocol 4 ensures resilience against ML-attacks, it has scalability issues. As discussed in Section III, all existing PUF-based protocols in literature use one of these approaches (shown in Protocol 1 through Protocol 4).

Next, we evaluate the systems performance of the above four protocols. For that, we consider a SRAM PUF that is implemented on a Xilinx XC5VLX30 with system clock of 1.84 MHz and 16KByte of program memory. In order to evaluate our noisy-PUF-based solutions (i.e., protocol 2, protocol 3, and protocol 4), before execution of each phase of the above protocols, we power cycle the device to re-initialize the SRAM-PUF. This gives us a real SRAM PUF noise profile. Next, for protocol 2 and protocol 3 we construct

Table I
COMPARATIVE ANALYSIS BASED ON THE DESIRABLE IMPERATIVE FEATURES (DIF)

Schemes	DIF1	DIF2	DIF3	DIF4	DIF5
Protocol 1	Yes	No	No	Yes	No
Protocol 2	Yes	Yes	No	Yes	No
Protocol 3	Yes	Yes	Yes	Yes	No
Protocol 4	Yes	Yes	No	No	Yes
DIF1: Mutual Authentication; **DIF2:** Handling Noisy Conditions**; DIF3:**Two-Factor Security;					
DIF4: Scalability; **DIF5:** Handling Machine-Learning or Modeling-Attacks;					

Table II
BENCHMARK ANALYSIS OF THE PUF-BASED SOLUTIONS

	Computation Cost (at the Device)	Execution Time (in clock cycles)
Protocol 1	P+2H+RNG	7,284+2×32,145+16,552 = **88,126**
Protocol 2	P+2H+FE.Rec+RNG	7,284+2×32,145+412,968 +16,552 **= 501,094**
Protocol 3	P+3H+FE.Gen+RNG	7,284+3×32,145+ 268,820+16,552 = **389,091**
Protocol 4	2P+FHD+RNG	2×7,284+11,670+16,552 = **42,790**
P: PUF Operation; **H:** Hash Operation (SHA-256); **FE.Gen:**Key Generation Algorithm;		
FE.Rec: Reconstruction Algorithm; **FHD:** Fractional Hamming Distance; **RNG:** Random Number Generation;		

the helper data from a (63,16,23)-BCH code [12]. From Table II, we can see that the Train-PUF-based solution (protocol 4) takes significantly less computation cost and execution time as compared to the others, since in this protocol the IoT device does not require to perform the computationally expensive FE.Rec, FE.Gen, and H (hash) operations but this construction suffers from the scalability issues. From Table II, we also note that the execution time of the protocol 2 is much higher than others. Also, in this construction the device needs store a set of pairs of the challenge-helper data pairs $(C, hd) = \{(C_1, hd_1, \cdots, (C_n, hd_n)\}$ that cause additional storage overhead at the device end, which could be an issue for resource limited IoT devices. Therefore, in a nutshell, it can be easily argued that the existing PUF-based authentication protocols for IoT devices suffer from various challenges (as shown in Table I and Table II).

F. Suitable Applications

The four protocol constructions have their relative strengths and may find applications based on the requirements of the underlying scenario. This section describes suitable IoT applications for the above four protocol. Since operating conditions (such as temperature variations) may influence the PUF operation, specially in terms of noise, Protocol 1 is not suitable for environments with varying conditions. However, Protocol 1 can be used in closed-IoT platforms such as smart-home environments, smart-door locks, etc. In protocol 2, the device needs to perform a computationally expensive Rec$(\cdot)$ function and also needs to store helper data. Hence, this construction is better suited for IoT devices with more computational power and storage capacity (e.g., smart phones). In Protocol 3, the device needs to maintain a secret key in order to avoid performing the computationally expensive Rec$(\cdot)$ function. Any loss of this key makes this construction insecure. Hence, this construction will be suitable for closed IoT platform devices such as smart-meters, smart tokens, etc. Finally, since scalability is the major issue in Protocol 4, this construction is better suited for IoT applications with limited number of devices such as IoT-based control-system applications, where the server may regularly check the status of a limited number of devices.

IV. OTHER CHALLENGES AND FUTURE EXPECTATION

Apart from the protocol level challenges described above, existing PUF-based security solutions for IoT systems may face the following issues in the aspects of efficiency and reliability.

1) PUFs have gained popularity in the security domain as an useful paradigm for hardware-based root of trust for resource constrained devices. However, the requirement of a specially manufactured IC for PUF-based devices (such as IoT device) is an issue since such hardware is not widely available at this moment. Therefore, despite of its advantages, PUF-based solutions are still not widely used in practice for security of IoT devices. As a result, these devices are often exposed to hardware attacks such as cloning and tampering of the devices. However, PUFs have received considerable interest in the industry recently (e.g., hardware and solutions from Xilinx, Intrinsic ID, Quantum Trace, ICTK, and Quantum Base [4]) and it is expected that more devices with in-built PUFs will be available in the near future.
2) The quality of existing PUFs is another concern, as it directly affects the reliability and security. Reliability of a PUF refers to its ability to operate correctly under a wide range of external circumstances and to have a sufficiently long lifetime. Operating conditions that influence PUF behavior include: *temperature*, *core voltage*, and *electromagnetic radiation*. On the other hand, the amount of randomness or entropy present in the

PUF is an important parameter for security. Achieving sufficient entropy in the PUF output is a challenging design problem and lack of entropy may cause breaches in PUF-based security solutions. There have have been many recent initiatives towards the design of high entropy PUFs (e.g., ring oscillator based PUFs [3]), and more solutions are expected in the near future.

3) Machine learning or modeling attacks have become one of the main security challenges for PUF-based security solutions. One solution to address this problem that has been explored in literature is through interesting PUF-constructions (such as [11]). However, most of them have been proven to be insecure [12]. On the other hand, although a few protocol level approaches have been proposed in literature (e.g., [15]), they are not scalable, making them unsuitable for IoT environments. Considering the ongoing research interest in this area, new PUF-constructions as well as new protocol-level solutions that are scalable and provide protection against ML-attacks can be expected in the near future.

V. Conclusion

This article presented the state-of-art approaches for designing PUF-based authentication protocols for resource constrained IoT devices. Protocols specific to ideal and noisy PUF conditions were presented. Moreover, a protocol level approach to deal with ML-attacks on PUFs was also presented. A comparative analysis was presented to show the effectiveness and weaknesses of of the four design approaches. Finally, the article discussed the challenges that PUF-based security solutions may face, and future expectations in this area.

References

[1] P. Gope, T. Hwang, "BSN-Care: A Secure IoT-based Modern Healthcare System Using Body Sensor Network," *IEEE Sensors Journal*, Vol. 16 (5), pp. 1368 – 1376, 2016.

[2] P. Ravikanth, "Physical One-Way Functions, " *Ph.D. thesis. Massachusetts Institute of Technology*, 2001.

[3] S. G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation" *in: Design Automation Conference*, 2007, DAC '07, 44th ACM/IEEE, 2007, pp. 9–14.

[4] T. McGrath, I. Bagci, Z. Wang, U. Roedig and R. Young, "A PUF taxonomy," *Applied Physics Reviews*, vol. 6, no. 1, 2019.

[5] J. R. Wallrabenstein, "Practical and secure IoT device authentication using physical unclonable functions" *In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud),* 76, pp. 99-106, 2016.

[6] P. Gope, J. Lee, and T. Quek, "Lightweight and practical anonymous authentication protocol for RFID systems using physically unclonable functions," *IEEE Transactions on Information Forensics and Security*, Vol. 13 (11), pp. 2831-2843, 2018.

[7] D. Moriyama, S. Matsuo, M. Yung, "PUF-based RFID authentication secure and private under complete memory leakage," IACR Cryptology ePrint Archive 2013, 712 (2013), http://eprint.iacr.org/2013/712.

[8] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.R. Sadeghi, A. R., I. Verbauwhede, I., and C. Wachsmann, C. "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs,," *In International Conference on Financial Cryptography and Data Security,* pp. 374-389, Springer, Berlin, Heidelberg, 2012.

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

[9] Y. Dodis, L. Reyzin, A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, " *In: Advances in Cryptology (EUROCRYPT)*. LNCS, vol. 3027, pp. 523–540 (2004)

[10] P-Ha Nguyen, D-P Sahoo, C. Jin, K. Mahmood, U. RÃŒhrmair, and M. Dijk, "The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks" *IACR Transactions on Cryptographic Hardware and Embedded Systems* pp. 243–290, August 2019.

[11] N. Wisiol et al. "Splitting the Interpose PUF: A Novel Modeling Attack Strategy, " *IACR Transactions on Cryptographic Hardware and Embedded Systems*, April 2020.

[12] J. Delvaux, D. Gu, I. Verbauwhede ,M. Hiller, "Efficient Fuzzy Extraction of PUF-Induced Secrets: Theory and Applications" *In: Cryptographic Hardware and Embedded Systems (CHES)* LNCS vol. 8913 pp. 412-430, Springer (2016).

[13] A. Vijayakumar, V. Patil, C.B. Prado, and S. Kundu, "Machine learning resistant strong PUF: Possible or a pipe dream?, " *In 2016 IEEE international symposium on hardware oriented security and trust (HOST)*, pp. 19-24, 2016.

[14] U. Ruhrmair, "PUF modeling attacks on simulated and silicon data," *IEEE transactions on information forensics and security,* 8(11), 1876-1891, 2013.

[15] M. Majzoobi , M. Rostami , F. Koushanfar , D.S. Wallach , S. Devadas, "Slender PUF protocol: A lightweight, robust, secure authentication by substring matching," *in proc, IEEE Symp. Security Privacy Workshops* pp. 33-44, 2012 .

[16] K. Yasunag, K. Yuzawa, "On the Limitations of Computational Fuzzy Extractors,". *ePrint Arch.* 2014, 3, 7-9.

[17] D. Nedospasov, J. P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF Analysis" *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC),* pages 30-38, 2013.