PUF-based mutual authentication and session key establishment protocol for IoT devices

Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa

Noname manuscript No.
(will be inserted by the editor)

PUF-based Mutual Authentication and Session Key Establishment Protocol for IoT Devices

Fahem Zerrouki · Samir Ouchani · Hafida Bouarfa

Received: date / Accepted: date

Abstract The Internet of things (IoT) is an indispensable part of our daily lives, bringing us many conveniences, including e-commerce and m-commerce services. Unfortunately, IoT networks suffer from several security issues, such as privacy, access control, and authentication. However, due to the limited computation resources, remote authentication between IoT devices and servers is vulnerable to being attacked over an insecure communication channel. Many authentication schemes have been proposed, but generally, they are based on traditional cryptographic techniques. Unfortunately, most of them are vulnerable to physical attacks since they rely mainly on a stored secret key in the device's local memory. However, recently Physically Unclonable Functions (PUFs) have been classified as solid security primitives that could guarantee the three pillars of security (confidentiality, authenticity, and privacy) of sent or received information by IoT devices. PUFs extract unique information from the physical characteristics of the IoT device. Nevertheless, a Fuzzy Extractor (FE) should be considered to extract correct and reproducible cryptographic keys from a noisy source. This paper proposes a mutual authentication and a session key establishment protocol for IoT devices based on Silicon PUFs using Arbiter chips. We also validate our developed protocol regarding its resistance to attack scenarios. By relying on formal verification using VerifPal, we found that the proposed authentication mechanism is secure and suitable for resource-constrained IoT devices. Furthermore, our scheme is more efficient than the existing ones in terms of attack robustness. Finally, the experiments have been validated on an Arbiter PUF dataset.

Keywords Physically Unclonable Functions · Authentication Protocol · IoT · Fuzzy Extractor · One-way Function · Formal Verification · Attacks

1 Introduction

Nowadays, a massive number of devices are connected, building the so-called Internet of Things (IoT) [24]. The latter is a network of interconnected devices, objects, or things equipped with network connectivity, sensors, and other essential electronic components that enable them to produce, collect, and exchange data about their environment. Such devices are widely used in various domains like smart homes, smart cities, smart transportation, smart grids, public health, agriculture, energy management, etc. It is projected that by the year 2025, around 75 billion IoT devices will be deployed [33]. Indeed, IoT applications have a wide range of advantages, making our lives easier, more innovative, more convenient, and more personalized by altering the way we carry out our daily duties. However, despite all the benefits of IoT, these devices face several security issues and hazards that must be continuously managed and maintained, including authentication, privacy, access control, and data collection and management [24].

In practice, wireless technology is the most commonly used communication means by IoT devices, but a secure network communication system requires a robust authentication protocol that provides secure

F. Zerrouki
E-mail: ze.fahem@gmail.com
Université Blida 1, Laboratoire LRDSI ,Faculté des Sciences, BP 270, Route de Soumaa, Blida, Algérie

S. Ouchani
E-mail: souchani@cesi.fr

H. Bouarfa
E-mail: hafidabouarfa@hotmail.com

transmission and sessions. The authentication of IoT devices refers to verifying the device identity and preventing malicious devices from accessing the IoT network. However, this process is considered as a key requirement and the main security challenge in IoT networks. Unfortunately, any flaw in the authentication process allows a compromised IoT device to gain access to the IoT trusted network and conduct unauthorized communication, inject false data, access confidential data, and launch dangerous attacks. Hence, secure transmission protects data during the transmission process by guaranteeing integrity, confidentiality, and non-repudiation of data using different encryption schemes [31].

Due to IoT device constraints, it is challenging to deploy conventional authentication protocols since they require more processing power, large memory, and high energy sources. Nevertheless, traditional cryptographic systems, such as public or symmetric key algorithms, store the secret keys or sensitive information on the volatile memory of the device, which makes them vulnerable to physical attacks such as invasive, semi-invasive or side-channel attacks [29]. An attacker can steal the stored secret key or make a full copy of the IoT system, then exploit it for an identity theft attack. Consequently, physical attacks are more convenient for IoT applications since devices can be found in public areas.

The discussed limitations and issues related to the IoT led to the emergence of lightweight authentication schemes by considering the specific nature and constraints of these devices. Therefore, any identification and authentication protocol designed for the IoT needs to be robust, computationally efficient, and secure against physical attacks. As a solution, physical unclonable functions (PUF) [29] have emerged as promising low-cost security primitives. More precisely, Silicon PUF [19] eliminate the need to store secret keys in device memory, making them a potential alternative to deploying more secure and low-cost authentication protocols for IoT systems.

1.1 Related work

Recently, many initiatives have proposed different authentication protocols for IoT devices. To overcome modelling and man-in-the-middle (MITM) attacks, Idriss et al. [14] proposed a lightweight PUF-based protocol that offers mutual authentication between IoT devices and the trusted server, where the device is equipped with a PUF circuit, and the server has a soft model of IoT PUF. With respect to the motivation of this work that targets lightweight protocols, Idriss et al. use extensive computational functions without considering the practical PUF reliability issues and the anonymity of devices.

Following the same idea of using PUF without storing CRPs on a trusted server, Kaveh et al. [16] proposed a two-way PUF-based authentication scheme for smart-grid communications. The specificity of this protocol is that it stores only one CRP on the trusted server "neighbourhood gateways", and at the end of each successful authentication phase, the device generates a new CRP to be stored on the server for future authentication. This protocol is incompatible with the basics of PUF, as smart meters store secret information and use it in the authentication process, making them vulnerable to physical attacks. Further, reducing some unnecessary variables could simplify the protocol. Yanambaka et al. [27] presented a PUF-based authentication protocol for the Internet of Medical Things (IoMT), using a secure database as a third entity where CRPs of IoMT devices are stored. In their proposed scheme, both the IoMT and the server were equipped with a PUF circuit. Unfortunately, this protocol does not guarantee reliability or anonymity. Also, no security analysis has been presented, and error correction has not been taken into consideration. In addition, the protocol is vulnerable to modelling attacks.

Compared to database-driven or server-based IoT authentication approaches, Zheng et al. [32] proposed a lightweight PUF-based mutual authentication and a key exchange protocol for Peer-to-Peer (P2P) or direct IoT connections. First, both IoT devices initiate an enrolment phase with a trusted server, where a set of information is generated, exchanged and stored on each IoT device with the help of the server. This protocol is designed to store information which makes it inadequate within the low-cost and resource-limited characteristics of IoT devices. Further, stored information allows physical attacks and makes this protocol impractical. If a connected device has to connect with a non-connected one, it must re-initiate a new enrolment phase. Thus, a DoS attack is inevitable.

Instead of using error correction alternatives such as fuzzy extractors to eliminate the noisy output of PUF, Najafi et al. [23] presented a device based authentication protocol without using error-correcting codes by using a deep convolutional neural network (CNN). The generated trained model of the device is stored on the trusted server to be used later to check if the received response from the device corresponds to the stored and used challenge. Unfortunately, the proposed scheme does not consider mutual authentication, anonymity and many type of attacks such as modelling, MITM, and denial of service (DoS).

Due to the importance of authentication of the broadcaster and the broadcasted message in smart grid networks, Ameri et al. [3] proposed a secure and broadcast authentication schemes for smart meters using a ring oscillator PUF. Manually, using mathematical proof, they showed that their proposed scheme can resist to physical attacks. However, their proposed protocol needs more memory and computing capacities with respect to smart meters devices. Similarly, Gope and Sikdar [11] proposed a privacy-aware authentication and key agreement scheme for secure smart grid communication, using PUF, one-way hash functions and BCH-based fuzzy extractor as error correction solution. The proposed protocol need to store data on the device memory including the public helper data, making this scheme vulnerable to physical and helper data manipulation attacks.

By combing the PUF output and user password as two-factor authentication, Guan et al. [12] proposed an identity PUF-authentication protocol for IoT devices. Unfortunately, the user is implicated in all authentication steps by receiving and transmitting data from the device to the server. Furthermore, the protocol does not take into consideration IoT constraints. Also, Mostafa et al. [21] proposed a lightweight mutual two-factor authentication mechanism between a device and a server. The proposed scheme uses the strong PUF for the authentication process, whereas the weak PUF generates a cryptographic key. This scheme does not present noise elimination, making it impractical in real applications.

Pu and Li [25] proposed a lightweight mutual authentication protocol between unmanned aerial vehicles (UAVs) and the ground stations while each UAV has its proper PUF. This protocol does not consider error correction, making it unpractical in a noisy environment. Following the same strategy, Kim et al. [17] proposed a PUF-based IoT device authentication protocol that does not take into consideration the noise elimination process, making the application of the proposed protocol impossible.

Aman et al. proposed a PUF-based authentication protocol called RapidAuth [2] that resists to physical attacks by using elliptic curve cryptography (ECC) as a second security primitive. Unfortunately, this scheme does not consider the noise elimination that plays a main role in the encryption and decryption of messages during the authentication steps. Muhal et al. [22] proposed a PUF-based authentication protocol. In fact, both the device and the server store an initial session secret key that will be used in the authentication phase. Unfortunately, storing this critical information on the device side allows physical attacks. Rather, the proposed protocol does not use any error correction and noise elimination techniques.

1.2 Motivation

As the presented related work schemes show, we ought to design an improved protocol that can efficiently guarantee a secure authentication scheme. From this perspective, our protocol seeks ways to address the following issues:

- The existing authentication algorithms are vulnerable to physical attacks as they have to store a piece of private information in the IoT device's small, low cost, and constrained resources.
- In a noisy PUF environment, the existing protocols do not ensure the integrity and confidentiality of messages.
- Protocols guaranteeing noise elimination suffer from helper data manipulation attacks.

1.3 Our Contributions

According to the reviewed initiatives, most protocols did not consider device-to-device authentication operations, rather than several known authentication protocol attacks, and more significantly, PUF helper data manipulation attacks.

In response to the above-discussed issues, this paper presents a lightweight device-to-device PUF-based authentication protocol and secure end-to-end communication between IoT devices for a peer-to-peer IoT connection. The trusted server ensures just the authentication step of the IoT device launching the communication. Our proposed protocol does not require storing secret cryptographic keys or other sensitive data in the IoT device's memory, making it resilient to physical attacks. The main contributions of this paper are summarized in a nutshell as.

1. Developing a mutual authentication protocol connecting IoT devices and a trusted server.
2. Establishing a secure session key management approach without relying on time consuming encryption algorithms.

3. Proposing a robust key derivation function such that the server dynamically computes a new session key from the PUF response of the IoT devices that attempt to communicate.
4. Guarantying the authentication in different environments and conditions by integrating the error correction techniques without storing any data in constrained devices.
5. Proving the security of the designed protocol by relying on formal methods techniques using Verifpal.

2 Preliminaries

2.1 Silicon PUF

A PUF is a robust hardware security primitive by exploiting the randomness that characterizes the physicality of PUF-based objects from which information can be extracted and then used in cryptographic applications. A Silicon PUF (SPUF) is a sub-class of electronic PUF, and due to manufacturing variations inherited in any fabrication process, an integrated circuit (IC) has a unique physical variation. Therefore, the behavior of each IC will be unique for the same given input. Exploiting this behavior, SPUF extracts unique information called "response" from an input called "challenge". A single challenge and its corresponding response form a so-called Challenge-Response Pair (CRP), which is used as a unique identifier or "fingerprint" of the IC or the device where it is embedded on. As illustrated in Fig. 1, given the same challenge for various PUF modules that are fabricated on the same IC, no two PUF modules will generate the same response [30].

Due to the manufacturing variations, it is impossible to get the same response from different challenges on the same PUF module. Industrially, SPUF is widely used in cryptographic key generation, device identification, and authentication. Uniqueness, uniformity, and steadiness are the three primary metrics used to evaluate the performance of PUF. According to the different sources of variation, SPUF is mainly divided into three major classes: Memory-based PUFs such as SRAM-PUF, delay-based PUFs like Arbiter PUFs, and Analog electronic PUFs such as Power distribution PUF [30].

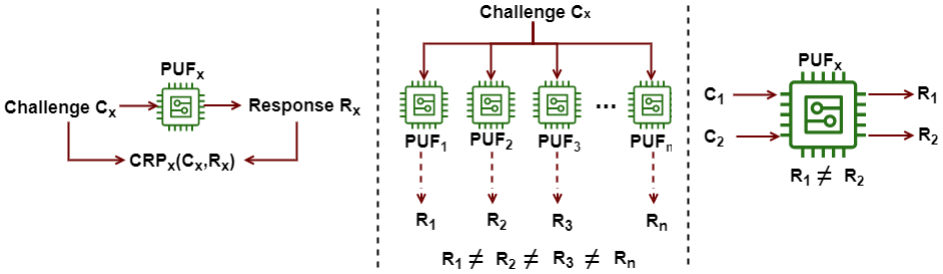

Fig. 1 The challenge response behavior.

Depending on the number of CRPs that could be generated from a given PUF, the latter can be classified into two categories weak or strong PUFs. A weak PUF is capable of generating a small number of CRPs, making them useful for deriving cryptographic keys. On the other hand, strong PUFs are characterized by a very large number of CRPs, making them more suitable for PUF-based authentication protocols.

2.2 Fuzzy Extractor

Intrinsically, PUF as a source of randomness can be used to generate cryptographic keys when needed. However, the changes in the environmental conditions directly affect the reliability of the PUF integrated silicon chips by causing noise on their outputs when challenged. Hence, for the same challenge on the same PUF module, the noise can cause errors in one or more PUF response bits, resulting in an unusable and incorrect response that differs from the original generated one. Therefore, the new response could not be used directly as a cryptographic key [28].

Fuzzy Extractor (FE) is one of the most commonly used coding techniques for response noise elimination and error correction. It is principally initiated to extract uniformly random strings from noisy and not uniformly random data. As shown by Fig. 2, FE consists of a pair of algorithms generation (Gen) and reproduction (Rep). Gen takes as input the initial response w and outputs a uniformly random string

R , which is used as a cryptographic key, and a non-secret string P (Public helper data). To reproduce R from the noisy response w', Rep algorithm takes two inputs: the public helper data P and w'. The reproduction process succeeds only if w and w' are close enough (w ≈ w') [28].

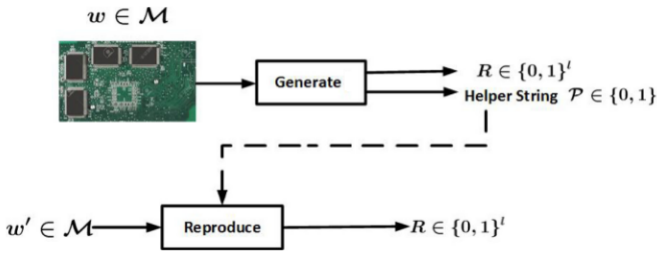

Fig. 2 Fuzzy Extractor.

2.3 Arbiter PUF (APUF)

A Delay PUF is a subclass of silicon PUF that exploits the propagation delay between the different delay paths of the PUF's circuits to generate a response. Arbiter PUFs are a widely used strong delay PUFs and the idea behind the arbiter PUF is to explicitly introduce a race condition between two digital paths on a silicon chip. It consists of the two delay paths as chains of switch blocks (multiplexers) and an arbiter block at the end of the chain. As shown in Fig. 3, the switch block has two possible configurations depending on the challenge bit; straight if the challenge bit is 0 and crossed if it is 1. Each switch block has two outputs and three inputs, two outputs from the previous stage and a single bit of the challenge. The inputs of the first switch block are connected to a common enable signal, and the outputs of the last switch block are connected to the arbiter block that determines which signal arrived first. Based on this result, the arbitrator makes one bit, which is called the response bit.

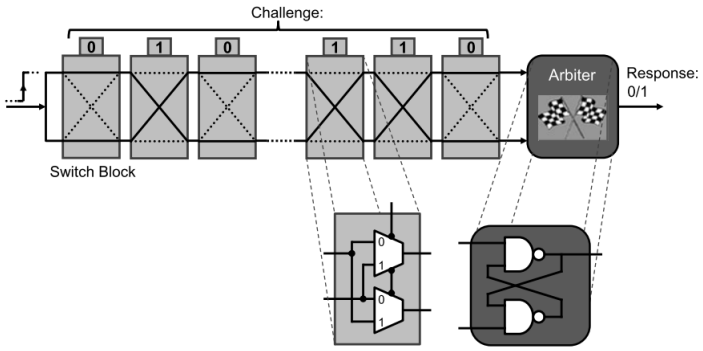

Fig. 3 The Structure of the Arbiter PUF.

2.4 Conventional PUF-based Authentication

To establish trust and secure communication in an untrusted network, PUF as a lightweight and robust security primitive, can be used to identify and authenticate IoT devices. Fig. 4 describes a conceptual PUF-based authentication process between an IoT device and a trusted server. PUF-based authentication protocols can be accomplished through two distinct phases. Firstly, during the enrolment phase, the server has access to the IoT device to apply a set of random challenges, then stores their corresponding sets of responses that are extracted from the PUF circuit integrated with the IoT device. The second phase is verification, in which the device verifies the identity of the IoT device. Next, the server randomly selects from its CRP database a challenge that has never been used. Then, the IoT device generates its corresponding response and sends it back to the server. If the response from the server matches the saved one for the challenge that was used, the IoT device is real and can connect to the IoT network.

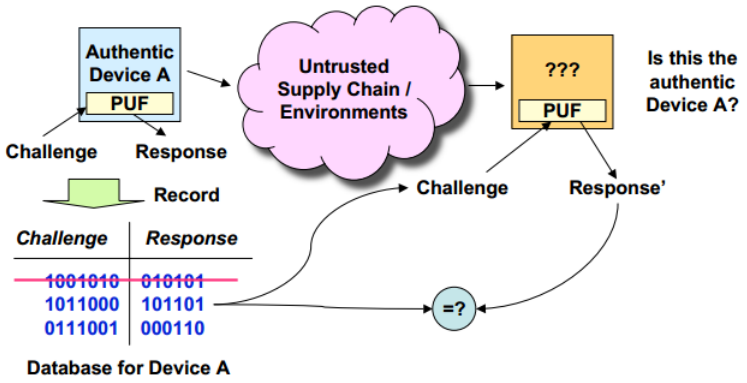

Fig. 4 A PUF-based Authentications Protocol Overview.

3 System Model and Security Model

This section presents the system and security models suitable for the proposed mutual authentication protocol. The system model shows the network architecture where protocols could be applied and the network component's requirements to operate and deploy our protocol. The security model involves threats found in the system model, including different attacks that an intruder can manipulate.

3.1 The System Model

As shown in Fig. 5, our network model looks like those used in [2, 21]. The network model contains mainly two entities: things and the server. Things could include various sensor devices, and the server is responsible for authenticating IoT devices and storing security parameters. IoT devices are assumed to communicate with the server through insecure public channels such as the Internet, making them vulnerable to diverse attacks. Although the considered system model is simple, the proposed protocol can be applied to various complex IoT network models. In this model, we assume that:

- IoT devices are considered highly resource-constrained devices with limitations in processing capabilities, memory, and energy.
- Each device is equipped with an integrated circuit consisting of an arbiter PUF, which gives the IoT device a unique identifier using the CRPs.
- Any tampering attempt to PUF will change the challenge-response data and yield the PUF incapable of generating the needed response to perform the authentication process. So, the IoT device fitted with a tempered PUF can never be authenticated as a member of the trusted network.
- The server is considered the trusted party with no limitation of resources and situated in a highly secured environment providing hardware/software network security solutions like IDS/IPS, firewalls, anti-flooding and anti-DoS, etc.
- The communication between the IoT device microcontroller and its PUF component cannot be accessed only through a secure channel.
- The enrollment phase is executed in a secure environment and through a secure channel.

3.2 Security Model

In the network model described above, we believe that IoT devices are untrustworthy, and that they could be installed in a public space without physical or hardware protection, allowing an adversary to capture them and obtain critical secrets from their local memory. Further, an intruder can eavesdrop, alter, and replay the exchanged messages during the authentication steps. Besides, the attacker cannot have access to the server's memory. More precisely, our security model considers a list of possible attacks that are realistic and were studied for PUF-based authentication protocols.

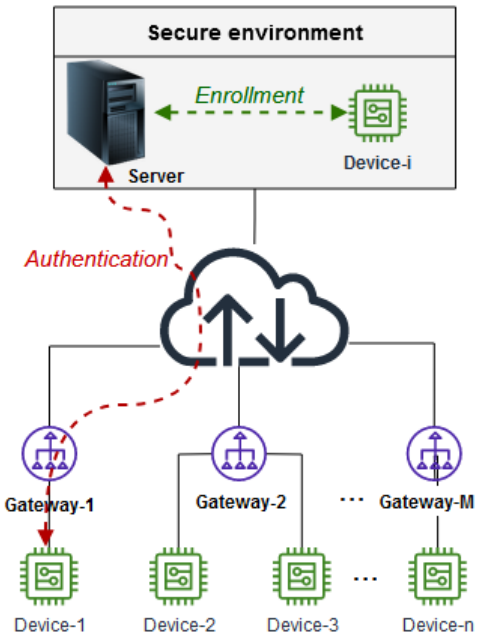

Fig. 5 The Internet of Things (IoT) system's model.

4 Proposed Scheme

In this section, we provide a detailed description of the proposed mutual authentication protocol between an IoT device equipped with an arbiter PUF and the server. A physical unclonable function, a hashing function, and a fuzzy extractor are employed in our scheme. The protocol consists of three modules; 1) the enrollment phase, 2) the authentication phase, and 3) session key establishment. The used symbols and cryptographic functions are defined in Table 1.

Symbols	Definitions
ID_A	The identity of a IoT device A
Reg_{req}	Registration request
$Auth_{req}$	Authentication request
$C_{A,i}$	The i^{th} challenge of the device A
h(.)	One-way hash function
PUF_A	The PUF of a device A
$R_{A,i}$	A response of the challenge $C_{A,i}$
$R'_{A,i}$	A noisy response of $C_{A,i}$
Gen(.)	Generation procedure of Fuzzy Extractor
Rep(.,.)	Reproduction procedure of Fuzzy Extractor
$K_{A,i}$	Extracted key from $R_{A,i}$
$P_{A,i}$	Helper data of $R_{A,i}$
TS	Timestamps
$\|\|$	Concatenation symbol

Table 1 Authentication protocol's symbols.

4.1 Enrollment Phase

In the most existing PUF-based authentication protocols that store a considerable number of CRPs in the authentication process the server challenges the IoT device with one randomly selected pair, and at the end, it will be deleted from the server database. In our proposed mechanism, the server stores only one pair. It minimizes the security threat due to the confidentiality of the stored data and the resources of the server database [17]. In this phase, when a new IoT device needs to be added as a member of the trusted network, it goes first with the server through the enrollment phase. This phase is executed in a trusted and secure environment. Fig. 6 describes the main steps of this phase to be performed by the *IoT-Device-A* and the $Trusted - server$ are as follows.

- *IoT-Device-A* sends its identity Id_A in plain text to the $Trusted-server$ with a registration request Reg_{req}.
- $Trusted-server$ generates randomly a challenge $C_{A,i}$, and sent it to *IoT-Device-A* within ID_A.
- The *IoT-Device-A* inputs this challenge into its arbiter PUF component to output the corresponding response $R_{A,i}$=$PUF_A(C_{A,i})$. Then, $IoT-Device_A$ sends to the server $ID_A, C_{A,i}, R_{A,i}$.
- Using the generation procedure of fuzzy extractor Gen, the $Trusted-server$ extracts the secret key $K_{A,i}$ and the public helper data $P_{A,i}$, ($K_{A,i}, P_{A,i}$)=Gen($R_{A,i}$). Then, the server computes the hash of the device identity and the secret key and stores h(ID_A), $C_{A,i}, h(K_{A,i}), P_{A,i}$ on its local secure database.
- In the end, $Trusted-server$ informs *IoT-Device-A* about the end of the registration process.

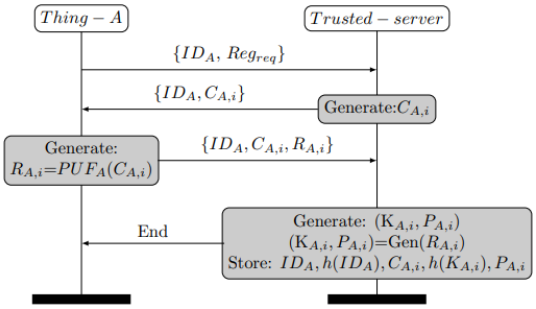

Fig. 6 Enrollment Phase.

After a successful registration, the Internet of Things device may be deployed and activated. Since preserved credentials ($C_{A,i}, h(K_{A,i}), P_{A,i}$) are automatically created at the conclusion of every successful authentication operation, the device will never seek for the server's permission to register.

4.2 IoT device authentication Phase

Our proposed mutual authentication scheme is based on deploying a silicon PUF, especially a strong arbiter PUF, in the IoT device. The idea behind using the PUF is that the IoT device uses the challenge-response pair of the PUF as a fingerprint and uses it to prove its identity with the server. The device and the server achieve mutual authentication since only those who know about the generated secret key for a given challenge in the enrollment phase and stored on the trusted server. In this proposed protocol, the server stores only one pair of CRPs to avoid attacks on the server. One of the most vital points of our protocol is that the IoT device does not store any secret or public information, which avoids physical attacks.

In the authentication phase to check the device's identity, PUF-based authentication protocols mainly compare the stored response in the enrollment phase to the new generated one to the same given challenge. Unfortunately, generating the same response for the same challenge in different environments and conditions such as voltage and temperature makes the response noisy or hazarded compared to the original one. This step is processed differently in our case, so to generate the secret key and store it safely on the server for the authentication process, the error correction technique has been adopted to eliminate the noise and ensure the comparison operation. More precisely, the proposed protocol takes into consideration the noise elimination process using the fuzzy extractor.

As shown in Fig. 7, the authentication process between the IoT device (*IoT-Device-A*) and the server (trusted-server) is running as follows.

1. Firstly, in **step (1)**, when the *IoT-Device-A* wants to communicate with the server, it generates a timestamp TS_1 and calculates a hash value of its identity (ID_A), $h(ID_A, TS_1)$. Then, it sends the hash of its identifier $h(ID_A)$, the authentication request $Auth_{req}$ and $h(ID_A, TS_1)$ message to the server.
2. In **Step (2)**, upon receiving the message from the IoT device, the server executes the following operations:
   - The server checks the existence of the received $h(ID_A)$ in its database.

- If the finding fails, the server rejects the authentication request. Otherwise, the server verifies the received message integrity by calculating $h(ID_A, TS_1)$ message and matches both the calculated hash messages within the received one.
- If the matching fails, the server rejects the authentication request. Otherwise, the server makes sure that the message has not been corrupted or tampered during the transmission phase. To calculate the message $h(h(ID_A), C_{A,i}, P_{A,i}, h(K_{A,i}), TS_2)$, the trusted server retrieves $C_{A,i}, P_{A,i}, h(K_{A,i})$ associated with the ID_A from its database and generates a timestamp TS_2.
- Finally, the server sends to the IoT device A: the stored challenge $C_{A,i}$, the helper data corresponding to this challenge $P_{A,i}$, TS_2, and the message $h(h(ID_A), C_{A,i}, P_{A,i}, h(K_{A,i}), TS_2)$.

3. In **Step (3)**, once the IoT device receives the server response, the following steps are executed.
   - The IoT device uses its arbiter PUF to generate the response of the received challenge: $R'_{A,i} = PUF_A(C_{A,i})$.
   - By default, the generated response is considered noisy. Then, the device reproduces the secret key $K_{A,i} = Rep(R'_{A,i}, P_{A,i})$ from the noisy response $R'_{A,i}$ using the reproduction process of the fuzzy extractor.
   - To verify the integrity of the received message from the server, *IoT-Device-A* calculates $h(h(ID_A), C_{A,i}, P_{A,i}, h(K_{A,i}), TS_2)$, and compares the received and the calculated hash messages. A successful matching provides the first factor for authenticating the server within the *IoT-Device-A*. This matching is ensured since the server is the only entity in the network that knows the secret key $K_{A,i}$ generated from the response $R_{A,i}$ of $C_{A,i}$.
   - If the comparison fails, the connection is rejected by the IoT device. Otherwise, the IoT device generates a timestamp TS_3, computes a new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$, and generates the response corresponding to the new computed challenge $R_{A,i+1} = PUF_A(C_{A,i+1})$.
   - Finally, the device calculates $h(h(ID_A), TS_3, k_{A,i}, R_{A,i+1})$ message, encrypts the new response with the reproduced secret key $(R_{A,i+1})_{h(K_{A,i})}$, and sends back the calculated message with TS_3 and the encrypted data.
4. Upon receiving the message from the IoT device, the server executes the following operations:
   - Decrypts $(R_{A,i+1})_{h(K_{A,i})}$ using the stored secret key $h(K_{A,i})$.
   - Verifies the integrity of the received data by calculating $h(h(ID_A), TS_3, k_{A,i}, R_{A,i+1})$.
   - Compares the received and the calculated hash messages. If the matching fails, the server terminates the connection. Else, the server validates the authenticity of the IoT device as a successful matching of these two hash messages. Consequently, the IoT is authenticated and the server communicates with the right IoT device.

4.3 Session key establishment

After authenticating the IoT device in the trusted server, both communicating entities can use the secret and exchanged key $h(K_{A,i})$ as a session key to secure the communication during the current session. $h(K_{A,i})$ is securely generated by the device and stored by the server. Also, it has never been used and communicated in plain-text. At the end of the communication, the server generates a new secret key and its corresponding helper data from the new generated response $R_{A,i+1}$, using the generation procedure of the fuzzy extractor $(K_{A,i+1}, P_{A,i}) = Gen(R_{A,i+1})$. Then, it calculates the new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$. Finally, the server updates the used information $C_{A,i}, P_{A,i}, h(K_{A,i})$ by the new one $C_{A,i+1}, P_{A,i+1}, h(K_{A,i+1})$ for a next authentication of the device ID_A.

5 Security Analysis

This section investigates the protocol's security against various types of attacks through informal and formal security analysis. In the first one, we check the robustness of the proposed protocol against the IoT known attacks, and in the second, we use Verifpal [18] as an automatic security verification tool, which is used to analyze the proposed protocol formally.

5.1 Informal security analysis

Our proposed mutual authentication mechanism is strong against the most existing attacks. This strength comes from using PUF without storing any information in the IoT device. Therefore, no secret value or

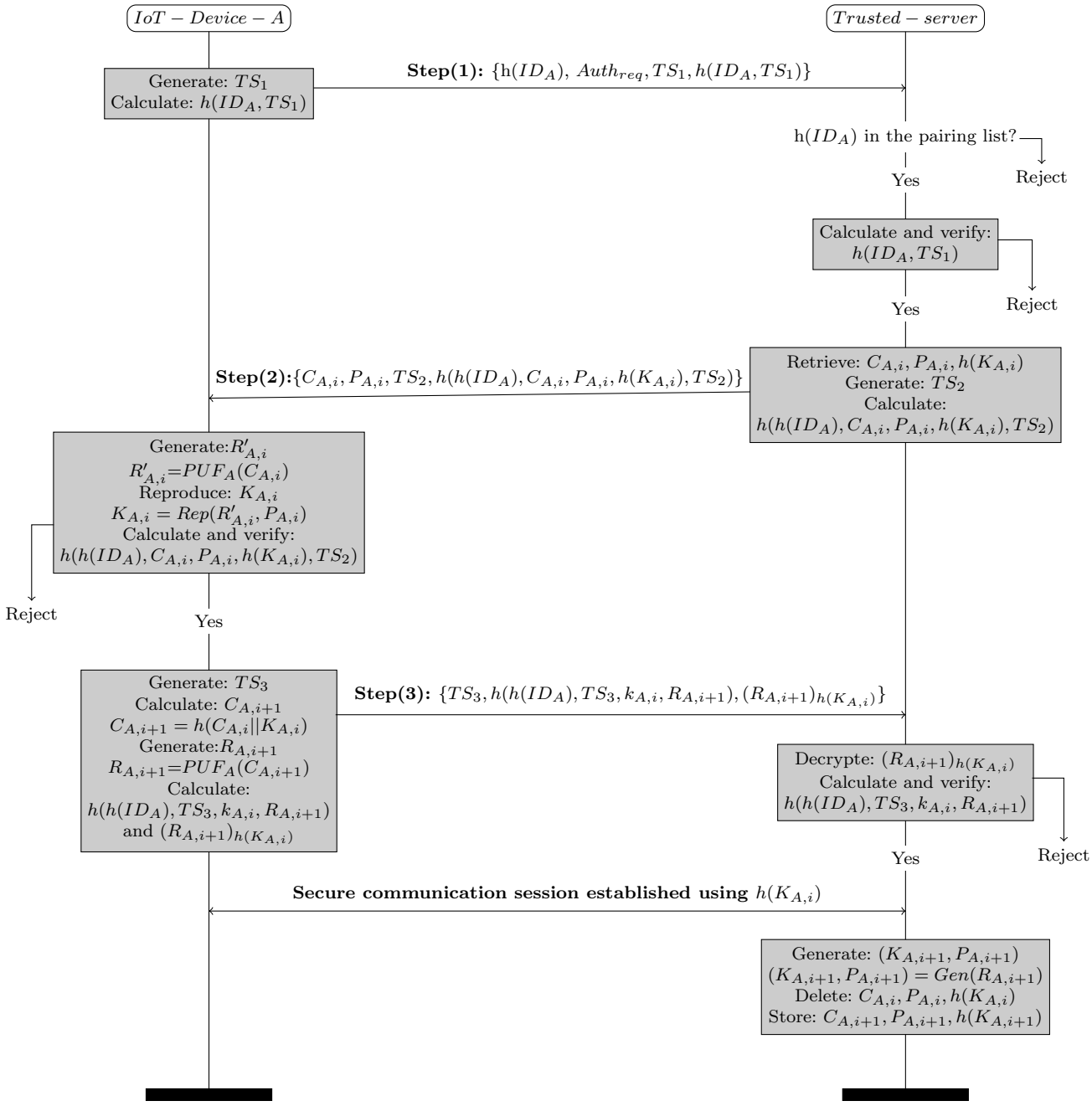

Fig. 7 Our proposed mutual authentication protocol.

key is stored or transmitted in the clear between the IoT device and the server. Using a fuzzy extractor as an error correction technique allows the generation and reproduction of secret keys that are used to encrypt the transmitted data during the authentication steps between the IoT device and the server. A defence assessment of each presented attack scenario is given in the following points.

- *Message Analysis Attack.* This kind of attack concerns the confidentiality of the exchanged message during or after a successful authentication phase. Meaning that an adversary tries to obtain the transmitted information between the communication entities. In our scheme, the exchanged data during the authentication steps is hashed by using a one-way hash function or encrypted using a secret key $h(k_{A,i})$. Also, the hashed values are not vulnerable to this attack, and the secret key $h(k_{A,i})$ is known only by the IoT device and the trusted server. Furthermore, the secret key changes after each session. Hence, this attack is not practical for our scheme.
- *Replay attack.* In this attack, the adversaries store the transmitted information of the valid authentication operation with the hope of utilizing it in future authentication. In our scheme, both the IoT

devices and the trusted server include timestamps in exchanged messages, which helps prevent this kind of attack. Besides that, the used information (challenge, response, helper data and secret key) changes in each authentication phase, and the session key also changes for each session. Hence, the attack will not be possible.

- *DoS attack.* The goal of adversaries in this attack is to exploit all the resources of the target device, and/or to temporarily or indefinitely disrupt services. It is accomplished by flooding the targeted device with superfluous authentication requests. In our proposed protocol, each entity verifies the received packets immediately by computing a hash value of the received message. If the integrity of the message is compromised, the attempted communication is banned. Further, except for the authentication request where the verification is based on the presence of the $h(ID_A)$ on the server's database, the random guessing of the hash value is not applicable as each hashed value contains the secret key $h(k_{A,i})$ which is known only by the server and the device equipped with the right PUF. Further, the server is installed in a secure environment, such as a data center, which can avoid and resist this kind of attack using software and hardware solutions. As a result, the proposed protocol has shown good resistance against DoS attacks.
- *Physical attack.* In this kind of attack, an adversary may attempt to physically get access to a device and try to obtain the secret information stored on its local memory. As mentioned before, our scheme is based on PUFs, which is a perfect solution against this kind of attack, where the keys are generated and reproduced through performing PUF operations, so there is no need to store the secret key. Further, in our scheme, the IoT device does not store any information in its local memory. Hence, the IoT device does not reveal any secrets even if an adversary has physical access, making the proposed protocol secure against physical attacks.
- *MITM and Injection attack:* In this attack, an adversary tries to intercepts the exchanged information between different entities during their authentication process. It possibly also alternating the communications between them and makes them believe that they are directly communicating with each other. In our authentication mechanism, the IoT device and the server do not exchange sensitive data (such as $h(k_{A,I})$, or the new response $R_{A,i+1}$) in plaintext over the unsecured communication channel. Also, an adversary has no access to these secrets to calculate the right hashed value for passing the verification step. So, the proposed protocol is secure against MITM attacks.
- *Modeling attack.* From the communication between the server and the device, an adversary tries to get CRPs of the used PUF. Then, it executes the machines learning techniques on the CRPs dataset and offers to the adversary the possibility of predicting the correct response related to a given challenge. In our mutual authentication scheme, we do not exchange the PUF response in plaintext, so if an adversary could capture the exchanged messages between the IoT device and the server, she/he could not run a machine learning attack since she/he possesses only the challenge that is exchanged in plaintext. Further, if an attacker has access to the trusted server, she/he can capture only one pair of CRPs. Therefore, launching a modelling attack on our proposed protocol is not applicable.
- *Helper data manipulation:* This attack targets to make the fuzzy extractor operations impossible by modifying the helper data transmitted or stored on the device memory. In our scheme it is securely generated by the server and transmitted to the IoT device. For each exchanged helper data, its hashed message with some other information are transmitted to help in verifying the integrity of the helper data on the IoT side. In addition, the manipulation can be executed only during the communication phase, where the helper data is transmitted to the IoT device. Consequently, a helper data manipulation attack is not feasible in our mutual authentication scheme.
- *Cloning attack:* Our proposed protocol resists cloning attacks since it exploits PUFs for authentication, and the main feature of PUF is unclonability as its name says. It has been shown that each PUF output is a unique value and it is hard to clone a PUF, due to the fact that any modification to the physical characteristics of the PUF circuit makes the cloned version a new instance of the PUF which results in generating different responses from the original PUF circuit.
- *Camouflage attacks:* In this attack, an adversary inserts into the network a counterfeit IoT device to operate as a normal device or attacks an authorized device in order to obtain, process, send or redirect and manipulate packets, or be passive and just analyze the traffic [1] . In our scheme, before adding the device to the network, it goes first through the enrollment phase with the server in a trusted environment, where the initial authentication credentials $(C_{A,i}, h(K_{A,i}), P_{A,i})$ were generated using embedded PUF component, and stored on the server. Further, the credentials are unique for each IoT device since they are extracted from the silicon PUF, which is a good solution for anti-counterfeiting [13]. So, even if an attacker inserts a forgery device equipped with a PUF, a camouflage attack cannot succeed as it does not figure out the server's database.

– *Anonymity Device:* In order to reveal the identity of the IoT device in the network and avoid the traceability of the device, the identity of the device is encrypted using a one-way hashing function that makes it difficult to recover the device identity from the hashing results.

With respect to the existing protocols, Table 2 compares our developed protocol to the recently deployed ones by considering their capabilities to avoid the above discussed attacks in Section 5.1. In the following comparison, Yes means that the protocol is not secure against the indicated attack. No means the inverse of the previous statements regarding the attacks. Also, **?** means that the authors of the cited contribution did not express any purpose regarding the selected attack.

		Replay Attack	Message Analysis Attack	Physical attack	DoS Attack	Modeling attacks	MITM attacks	Cloning attack	Injection attack	Helper data manipulation	Anonymity Device	Anonymity server
Works	Idriss et al. [14]	No	Yes	No	No	No	No	No	No	?	No	No
	Kaveh et al. [16]	No	No	Yes	No	No	No	No	No	Yes	Yes	No
	Yanambaka et al. [27]	Yes	Yes	No	Yes	Yes	Yes	No	Yes	?	No	No
	Zheng et al. [32]	No	No	Yes	No	No	No	No	No	Yes	No	No
	Najafi et al. [23]	Yes	Yes	No	Yes	Yes	Yes	No	?	?	No	No
	Guan et al. [12]	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	**?**
	Pu and Li [25]	No	No	No	No	No	No	No	No	?	No	?
	Meng et al. [20]	No	No	No	Yes	No	No	No	No	No	No	?
	Aman et al. [2]	No	No	No	No	No	No	No	No	No	No	No
	Kim et al. [17]	No	No	No	Yes	No	No	No	No	?	No	?
	Muhal et al. [22]	No	No	Yes	No	No	No	No	No	?	No	No
	Mostafa et al. [21]	No	No	No	No	No	No	No	No	?	No	?
	Ameri et al. [3]	No	No	No	No	No	No	No	?	?	Yes	No
	Gope and Sikdar [11]	No	No	No	Yes	No	No	No	No	Yes	?	No
	Proposed scheme	No	No	No	No	No	No	No	No	No	No	No

Table 2 Security of the proposed PUF-Based authentication Protocols.

5.2 Formal security analysis

Before implementation, we first check the correctness and the robustness of our proposed mutual authentication protocol using a formal security analysis. In this phase, we rely on the symbolic verification tool, Verifpal. Compared to the most relevant protocol verification tools based on symbolic models, mainly Tamarin and ProVerif, Verifpal extends them to cover more properties to construct correctly and verify efficiently cryptographic communication protocols, especially the following.

– *Modeling* looks if graphical or pre-defined behaviours can be instantiated and repeated.
– *Cryptography* shows if cryptographic syntax is supported like standard hashing, encryption and decryption functions.
– *Unbound* is to not limit the number of communication sessions.
– *Equational* presents if the user defined functions are supported, in addition to classical communication calculus properties, like: associative and commutative properties.
– *Mutability* shows if the tool support the verification of protocols with global mutable state.
– *Equivalence* checks if the properties related to equivalence checking are enabled, like: trace equivalence, bisimilarity, etc.
– *Implementation* shows how possible is generating an executable code.

The comparison presented in Table 3 extending the results presented in [4] shows that Verifpal [18] is a good candidate to specify correctly and verify efficiently cryptographic protocols.

Tool	Modeling	Cryptography	Unbound	Equational	Mutablity	Equivalence	Implementation
CPSA [9]	○	○	●	○	●	○	○
DEEPSPEC [7]	○	◐	○	◐	●	●	○
F7 [5]	○	◐	●	◐	●	○	●
Maude-NPA [10]	○	◐	●	●	○	●	○
ProVerif [6]	◐	◐	●	◐	○	●	○
Scyther [8]	◐	○	●	○	○	○	○
Tamarin [26]	◐	◐	●	●	●	●	○
Verifpal [18]	●	●	●	◐	●	◐	◐

Table 3 Comparison between Verifpal with the main symbolic security analysis tools [18].

To describe our protocol model in Verifpal, we need to define whether the model will be analyzed under a passive or active attacker. In our case, we chose as follows an active attacker who can modify the intercepted messages and inject messages during the authentication phase.

```
attacker[active]
```

Secondly, we define the different principals taking part of the protocol model rather than the attacker. For simplicity, we consider only two entities, the `IoT_device_A` and the `Trusted_server`.

```
principal IoT_device_A [...]
principal Trusted_Server [...]
```

Then, we need to describe the manipulated data and the executed operations by each entity. In the following Verifpal listing, `Knows blueprivate` indicates that `ID_a` is classified as a private information known only by `IoT_device_A`. Then, `generate` is used to create a unique timestamp `TS_1`, `HASH` returns the hashing values of the tuple `ID_a and (ID_a, TS_1).`

```
principal IoT_device_A [
    Knows private ID_a //IoT device A identity
    generate TS_1 //Timestamps_1
    hash_ida=HASH(ID_a) //Calculate ID_a hashing value
    msg_1=HASH(ID_a, TS_1) //Calculate (ID_a, TS_1) hashing value
    ]
```

After that, we illustrate the messages being communicated between the principals across the network. As an example, the following listing shows the message composed from `hash_ida`, `TS_1`, and `msg_1` that is sent in **Step (1)** by `IoT_device_A` and received by `Trusted_Server`. The whole Verifpal code of the proposed scheme is presented in Section 9.

```
IoT_device_A -> Trusted_Server : hash_ida, TS_1, msg_1
```

Lastly, we request Verifpal the following queries about the integrity, privacy, and freshness of the messages exchanged and the authentication of the participants.

- **Integrity:** In each step, after receiving the message by the device or the server, we check the integrity of the received packet by computing its hash function. To check this property, we use `ASSERT` a Verifpal's predefined primitive. `msg_1` checking integrity using Verifpal is illustrated as follows.

```
ASSERT(HASH(ID_a, TS_1),msg_1)?
```

- **Confidentiality:** In our scheme `ID_a, K_A_i, K_A_ii` are considered as confidential information for the attacker, and are known only by the principals: the IoT device and the server. To test this property, we declare first the private information using `Knows private` predefined primitives, and the secret information is declared as follows.

```
    Knows private ID_a, K_A_i, K_A_ii
```

Then, we ask Verifpal the following confidentiality query of the private declared information. This query tests if an attacker could obtain the private information communicated across the network

during the authentication phase. One example of these queries is expressed as follows to check the confidentiality of the secret key `K_A_i`.

```
confidentiality? K_A_i
```

- **Mutual Authentication:** After describing the transmitted messages during the authentication phase, we use the authentication query of Verifpal to check the mutual authentication process between the IoT device and the server. As example, the next query tests if the `IoT_device_A` could authenticate the `Trusted_Server` based on the received message `msg_2`.

```
authentication?Trusted_Server ⟶ IoT_device_A: msg_2
```

- **Freshness:** This property checks the freshness of the transmitted messages. It helps to detect replay attacks, where an attacker could manipulate one message, and send it at a different time. An example of testing the freshness of msg_2 using Verifpal's freshness query is the following.

```
freshness? msg_2
```

The verification results of the formal security analysis of the proposed protocol are shown by the execution of the Verifpal code given in Section 9. Fig. 8 shows that all the quires were passed successfully. This means that it is infeasible for active attackers to decrypt messages without secret keys and either to get the secret session key used between the IoT device and the trusted server. Further, an intruder cannot launch MITM and injection attacks due to the freshness of the timestamp value. Finally, both the IoT device and the trusted server can authenticate each other based on the secret key generated using PUF. Consequently, the mutual authentication protocol using PUF is secure and guaranteed.

```
Verifpal • Verification completed for 'PUF-based-Protocol.vp' at 01:00:47 PM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
```

Fig. 8 The verification results of proposed scheme using Verifpal.

6 Performance Evaluation

In this section, the performance evaluation of the proposed protocol is analyzed in terms of computational complexity, communication overhead, and storage constraints. We compare our scheme to Gope and Sikdar [11] and Kaveh et al. [16] since they consider our protocol's features like mutual authentication and error correction.

6.1 Computational Complexity

As illustrated in Table 4, the computational complexity consists of the primary operations and their frequency involved in completing the authentication process. In the proposed schemes, the main used operations are hashing, encryption/decryption, and PUF operations. Table 4 shows the time required by the IoT device and the trusted server to achieve the authentication steps of our proposed mechanism, Gope and Sikdar scheme [11] and Kaveh et al. scheme [16] . The hashing time is denoted by N_H and N_{PUF} is the time for a PUF to produce a response, N_{ENC} is the time needed for encryption/decryption operations, and N_{FUZZ} represents the time of fuzzy extractor operations. We measure these values by counting their number of occurrences from Fig. 7.

Let's assume the use of a one-way hashing function such as *SHA-2* that has a worst-case running time of O(n), where n is the number of bits in the Hashing operation's outputs. Also, by using block ciphers, the average encryption and decryption times vary with respect to the key length, and its complexity can be considered O(n).

As illustrated in Fig. 9 and Fig. 10, both compared schemes have the same number of PUF and fuzzy extractor operations. The proposed protocols have a complexity of O(n) for the IoT devices as well as the

Works	IoT Device	Server
Gope and Sikdar	5 N_H + 1 N_{FUZZ} +1 N_{PUF} +3 N_{ENC}	6 N_H+1 N_{FUZZ} + 3 N_{ENC}
Kaveh et al.	5 N_H + 1 N_{FUZZ} +2 N_{PUF}+3 N_{ENC}	5 N_H+1 N_{FUZZ}+3 N_{ENC}
Proposed work	5 N_H + 1 N_{FUZZ} +2 N_{PUF} +1 N_{ENC}	4 N_H+1 N_{FUZZ} +1 N_{ENC}

Table 4 Computational complexity comparison.

server. Thus, the computational complexity of our proposed mechanism is lower since hashing, encryption, and decryption operations in the presented schemes in Table 4 are higher than in our proposed one.

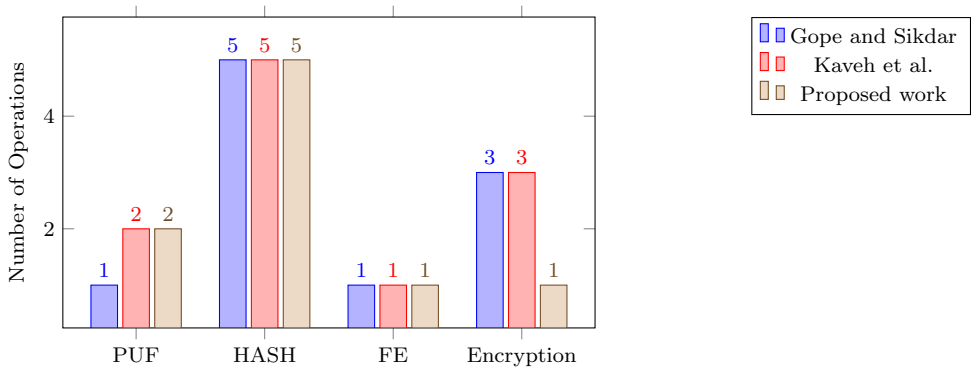

Fig. 9 The Computational Complexity on the device side.

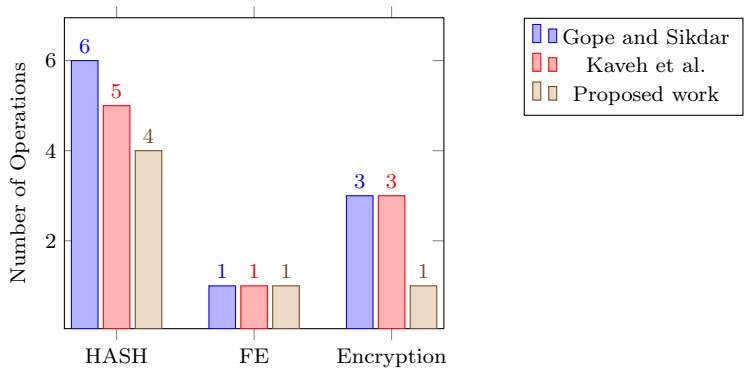

Fig. 10 The Computational Complexity on the server side.

6.2 Communication Overhead

The communication overhead corresponds to the total number of transmitted bytes between the IoT device and the server during the authentication phase. To calculate the length of each transmitted message, we use the message parameters that are communicated along with their sizes given in Table 5.

Based on the values in Table 5 and the protocol steps in Fig. 7, we infer that the transmitted bytes in **step (1)**, **step (2)**, and **step (3)** of the authentication phase are 70, 78 and 54, respectively, which results-in 202 transmitted bytes between the IoT device and the server. The communication overhead of Kaveh et al.'s protocol is 248 bytes. As a result,the communication overhead in our proposed mutual authentication scheme is lower than Kaveh et al.'s mechanism by 18.55%.

6.3 Storage Constraints

Compared to the most surveyed IoT PUF-based mutual authentication protocols that store a large number of CRPs on the server-side for each device, our proposed mutual authentication protocol is very efficient

Message Parameters	Size in Bits
$Auth_{req}$	1
Hash function	256
Timestamps TS1, TS2, and TS3	48
Challenge	128
Helper data	64
Encrypted data	128

Table 5 Authentication messages' parameter values.

in terms of storage requirements. Rather than the IoT device does not store any secret or non-secret information, the server keeps only one CRP pair for each device's new authentication which makes our mechanism more scalable for a large number of IoT devices that could be deployed in the system. After establishing a secure connection, the server deletes the data used during the authentication process except for the device identity. Contrarily, the IoT devices in [16, 11] have a storage constraint: they have to keep secret information in their local memory.

7 Experiments

To check the integrity of the exchanged data and verify their authenticity, our scheme reinforces the use of PUF and FE operations by including one-way hashing algorithms that have high resistance against cryptanalysis. In practice, we propose using the SHA-2 hash standard algorithm in our proposed mechanism since it is implemented in well-known security applications such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL). Further, it has been implemented in many security ICs for commercial purposes. Compared to SHA-3, SHA-2 has more hardware and software support which makes it easier and faster. For the encryption and decryption operations, we assume the use of block ciphers with 128-bit keys such as CLEFIA [15] and AES [21].

To show the effectiveness and applicability of our scheme, we consider a critical system in which the IoT end device collects sensitive data that needs to be transmitted to a trusted server that stores data in a secure way. In the current experiment, the end device is equipped with an Arbier PUF module and has a communication capabilities. For that, we use the APUF dataset [1] generated from an 16-bit Arbiter-PUF, containing 30 instances with 30 responses of output 32 bits under 25C. To show the noise effect, the same challenge is given as input under two different temperatures: -40C, and 85C. Table 6 illustrates the generated response by the fourth APUF instance. By considering $R_{4,1}$ as the initial response, bits in red corespondents to the error bit in the new generated responses $R'_{4,1}$ and $R''_{4,1}$.

APUF#3	25C : $R_{4,1}$	01010000101001111111110010101111
	-40C : $R'_{4,1}$	01010000101001111111111010101111
	85C : $R''_{4,1}$	00010000101001111111110010001111

Table 6 APUF instance 4 response under different temperatures.

7.1 Arbiter PUF Evaluation

We show how much this APUF is stable by evaluating the reliability metrics. It calculates the consistency between instances' outputs across environmental variations such as temperature. For a given PUF instance $APUF_i$ has the response R_{d,m_1} of P-bit reference at normal operating conditions m_1 and the response R_{d,m_2} of P-bits at different conditions m_2 for the same challenge c, we have:

$$Reliability = 1 - \frac{1}{RMP} \sum_{m_2=2}^{M} \sum_{p=1}^{P} (r_{p,d,m_1} \oplus r_{p,d,m_2})$$

[1] https://github.com/salaheddinhetalani/PUF

Fig. 11 shows the reliability of the used APUF which is between 93% and the ideal value 100%. This means that all 30 APUF instances generate close stable responses at different temperatures 25C, -40C, and 85C. Using the same challenge as an input, this APUF outputs a noisy response that has some error bits, compared to the reference. Contrarely, the unreliability is expressed by the expected bit error rate (BER) between two evaluations R and R' of the same response with $32-bits$. By considering the responses generated under 25C as reference (R), Fig. 12 shows and compares the BER values for the responses' evaluations for both $(25C, -40C)$ (blue color) and $(25C, 85C)$ (red color). For the first evaluation, 22 times BER is 0, which means that the output of APUF under -40C is stable in 22 cases. Also, the 8 other outputs were slightly lower than 3%, which means this helps to recover them more efficiently.

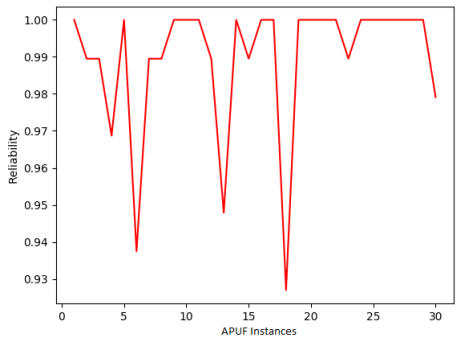

Fig. 11 APUF Reliability.

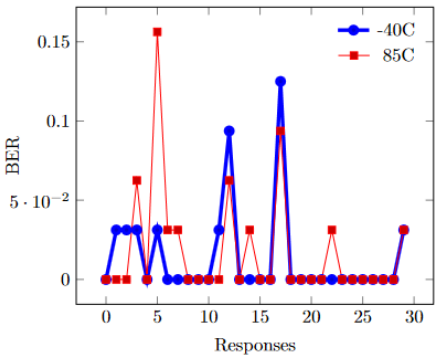

Fig. 12 Bit Error Rate Evaluation.

7.2 Enrollment Phase

Before installing a device, it goes first through the initialization setup phase where the initial credential information is generated and stored on the server. As an example, let's take the fourth APUF instance as the embedded PUF in the IoT device. After challenging the APUF with a 16-bits challenge, the server extracts the secret key and the helper data from the generated response $R_{4,1}$: `0101000010100111111111010101111`. To generate a secret key, the server uses SHA-256, and for the helper data, we implement the Fuzzy Extractor using a *syndrome construction*. We assume the use of the linear binary code block (6,3,1) as an error correction code block. At the end of the enrollment phase, in addition to the used challenge and the device identity, the server stores the extracted secret key $h(K_{4,1})$: `CE451782B639A8848BEC7564D596F5968852FF7FEC264DE5B14F15D7D2A862B5` and $P_{4,1}$: `(100,010,011,001,011,110)`. Finally, the IoT device equipped with the fourth APUF could be installed at its location.

7.3 Authentication phase

When the IoT device collects sensitive data, it needs to transmit it to a trusted server for storage. Thus, it needs to be authenticated by the trusted server. Similarly to the example presented in the enrollment phase, the transmitted data during the authentication phase is exchanged as follows.

- **Step (1).** Lets assume that the device identity is its MAC address as example `00:00:5e:00:53:af`, and the timestamp value is `:10-01-22 12:00:17`. The Final transmitted data in this step is: `7E5DC672DE56E106E5DE98B6F282A046377C4391FD6A3EF07507311211CB830F,1,10-01-22 12:00:17,B692B1241F28D0DF2F9B3504D36330D3B65D52585510045ECB516248248BAF3D`, where the second parameter indicates that it is an authentication request.
- **Step (2).** After verifying that the identity of the device and the integrity of the received data are good. The server challenging the IoT device by sending the next message: `0100010011010100101,(100,010,011,001,011,110),01-02-22 12:01:48,13E29C3A59A793B74B3DAC3B1CD75012E9890F34F717ED6733F25B4471DE3DD8`.
- **Step (3).** By receiving the previous data, the IoT device uses its APUF to regenerate the needed response to the received challenge. We assume that the new response is: `0101000010100111111111101010111`, which is generated under a -40C with one bit error. Then, the device launches the Fuzzy extractor process to correct the found error in this response, using the received syndrome `(100,010,011,001,011,110)` as the helper data. After the reproduction process, the integrity of the received

data is checked, and the session key is approved. Also, the device generates a new response, and encrypts it before sending it to the server.

Finally, the server decrypts and checks the validity of the received encrypted data. Then, it replaces the existing credential data with the new generated ones. At this stage of session key establishment, mutual authentication has been achieved from both sides (device and server) and they can communicate securely using the secret key `CE451782B639A8848BEC7564D596F5968852FF7FEC264DE5B14F15D7D2A862B5`.

8 Conclusion

In this paper, we have developed a PUF-based mutual authentication and session key establishment protocol for IoT devices that exploits the randomness of the Arbiter PUF. We have used a fuzzy extractor technique to generate a secret key and have performed noise elimination from the generated response. The proposed authentication protocol does not need to store any information on the local device memory, which avoids many attacks, especially physical-based ones. By relying on formal and informal security analysis, we proved that our developed protocol is secure against the most existing protocols, especially physical attacks. As perspectives, we intend to extend this work by:

1. Deploying the proposed protocol with a decentralized network architecture (blockchain) that exploits a PUF based on IoT and smartphones devices.
2. Considering device to device secure communication without relying on a server.
3. Measuring possible attack surfaces, energy consumption and communication costs.
4. Classify the possible failures related to our scheme and studying its reliability and resiliency.
5. Applying the protocol on autonomous vehicles and smart living systems.

References

1. Ahmed HI, Nasr AA, Abdel-Mageid S, Aslan HK (2019) A survey of iot security threats and defenses. International Journal of Advanced Computer Research 9(45):325–350
2. Aman MN, Chaudhry SA, Al-Turjman F (2020) Rapidauth: Fast authentication for sustainable iot. In: International Conference on Forthcoming Networks and Sustainability in the IoT Era, Springer, pp 82–95
3. Ameri MH, Delavar M, Mohajeri J (2019) Provably secure and efficient puf-based broadcast authentication schemes for smart grid applications. International Journal of Communication Systems 32(8):e3935
4. Barbosa M, Barthe G, Bhargavan K, Blanchet B, Cremers C, Liao K, Parno B (2021) Sok: Computer-aided cryptography. In: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, pp 777–795
5. Bengtson J, Bhargavan K, Fournet C, Gordon AD, Maffeis S (2011) Refinement types for secure implementations. ACM Transactions on Programming Languages and Systems (TOPLAS) 33(2):1–45
6. Cheval V, Blanchet B (2013) Proving more observational equivalences with proverif. In: International Conference on Principles of Security and Trust, Springer, pp 226–246
7. Cheval V, Kremer S, Rakotonirina I (2018) Deepsec: deciding equivalence properties in security protocols theory and practice. In: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, pp 529–546
8. Cremers CJ (2008) The scyther tool: Verification, falsification, and analysis of security protocols. In: International conference on computer aided verification, Springer, pp 414–418
9. Doghmi SF, Guttman JD, Thayer FJ (2007) Searching for shapes in cryptographic protocols. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, pp 523–537
10. Escobar S, Meadows C, Meseguer J (2009) Maude-npa: Cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V, Springer, pp 1–50
11. Gope P, Sikdar B (2018) Privacy-aware authenticated key agreement scheme for secure smart grid communication. IEEE Transactions on Smart Grid 10(4):3953–3962
12. Guan Z, Liu H, Qin Y (2019) Physical unclonable functions for iot device authentication. Journal of Communications and Information Networks 4(4):44–54
13. Hu YW, Zhang TP, Wang CF, Liu KK, Sun Y, Li L, Lv CF, Liang YC, Jiao FH, Zhao WB, et al. (2021) Flexible and biocompatible physical unclonable function anti-counterfeiting label. Advanced Functional Materials p 2102108

14. Idriss TA, Idriss HA, Bayoumi MA (2021) A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices. IEEE Access
15. Katagi M, Moriai S (2011) The 128-bit blockcipher clefia. IETF RFC 6114
16. Kaveh M, Aghapour S, Martin D, Mosavi MR (2020) A secure lightweight signcryption scheme for smart grid communications using reliable physically unclonable function. In: 2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), IEEE, pp 1–6
17. Kim B, Yoon S, Kang Y, Choi D (2019) Puf based iot device authentication scheme. In: 2019 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, pp 1460–1462
18. Kobeissi N, Nicolas G, Tiwari M (2020) Verifpal: cryptographic protocol analysis for the real world. In: International Conference on Cryptology in India, Springer, pp 151–202
19. Lim D, Lee JW, Gassend B, Suh GE, Van Dijk M, Devadas S (2005) Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13(10):1200–1205
20. Meng J, Zhang X, Cao T, Xie Y (2021) Lightweight and anonymous mutual authentication protocol for iot devices with physical unclonable functions
21. Mostafa A, Lee SJ, Peker YK (2020) Physical unclonable function and hashing are all you need to mutually authenticate iot devices. Sensors 20(16):4361
22. Muhal MA, Luo X, Mahmood Z, Ullah A (2018) Physical unclonable function based authentication scheme for smart devices in internet of things. In: 2018 IEEE International Conference on Smart Internet of Things (SmartIoT), IEEE, pp 160–165
23. Najafi F, Kaveh M, Martín D, Reza Mosavi M (2021) Deep puf: A highly reliable dram puf-based authentication for iot networks using deep convolutional neural networks. Sensors 21(6):2009
24. Nandy T, Idris MYIB, Noor RM, Kiah LM, Lun LS, Juma'at NBA, Ahmedy I, Ghani NA, Bhattacharyya S (2019) Review on security of internet of things authentication mechanism. IEEE Access 7:151054–151089
25. Pu C, Li Y (2020) Lightweight authentication protocol for unmanned aerial vehicles using physical unclonable function and chaotic system. In: 2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN, IEEE, pp 1–6
26. Schmidt B, Meier S, Cremers C, Basin D (2012) Automated analysis of diffie-hellman protocols and advanced security properties. In: 2012 IEEE 25th Computer Security Foundations Symposium, IEEE, pp 78–94
27. Yanambaka VP, Mohanty SP, Kougianos E, Puthal D (2019) Pmsec: Physical unclonable function-based robust and lightweight authentication in the internet of medical things. IEEE Transactions on Consumer Electronics 65(3):388–397
28. Zerrouki F, Ouchani S, Bouarfa H (2021) A generation and recovery framework for silicon pufs based cryptographic key. In: International Conference on Model and Data Engineering, Springer, pp 121–137
29. Zerrouki F, Ouchani S, Bouarfa H (2021) A low-cost authentication protocol using arbiter-puf. In: International Conference on Model and Data Engineering, Springer, pp 101–116
30. Zerrouki F, Ouchani S, Bouarfa H (2022) A survey on silicon pufs. Journal of Systems Architecture p 102514
31. Zhang J, Rajendran S, Sun Z, Woods R, Hanzo L (2019) Physical layer security for the internet of things: Authentication and key generation. IEEE Wireless Communications 26(5):92–98
32. Zheng Y, Liu W, Gu C, et al. (2021) Puf-based mutual authentication and key-exchange protocol for peer-to-peer iot applications
33. Zhou Q, He Y, Yang K, Chi T (2021) 12.3 exploring puf-controlled pa spectral regrowth for physical-layer identification of iot nodes. In: 2021 IEEE International Solid-State Circuits Conference (ISSCC), IEEE, vol 64, pp 204–206

9 Verifpal Code

```
attacker[active]
principal IoT_device_A[
    knows private ID_a //IoT device A identity
    generates TS_1 //Timestamps_1
    hash_ida=HASH(ID_a) //Calcule Hach(ID_A)
    msg_1 =HASH(ID_a, TS_1) //Calcule controle integrity message (1)
]
IoT_device_A -> Trusted_Server: hash_ida,TS_1,msg_1 // send message(1)
principal Trusted_Server[
    knows private  ID_a// Identity-hashed
    // verify if the recivied identy exists in the server's database
    _ =ASSERT(hash_ida,HASH(ID_a))?
]principal Trusted_Server[
    _ =ASSERT(HASH(ID_a, TS_1),msg_1)?// Check the integrity of the message (1)
]principal Trusted_Server[
    knows public C_A_i  //Challenge
    knows private K_A_i //Stored secret-key
    hash_ida_s=HASH(ID_a)
    generates TS_2 //Timestamps_2
    msg_2=HASH(C_A_i,TS_2,K_A_i)//Calculte integrity of message(2)
]//Send the data to challenge the IoT device
Trusted_Server -> IoT_device_A : hash_ida_s,TS_2,msg_2 //(2)
principal IoT_device_A[
    knows private K_A_i //EXtracted secret-key using PUF
    //check msg_2 integrity (Server authenticated ?)
    _ =ASSERT(HASH(C_A_i,TS_2,K_A_i),msg_2)?
]principal IoT_device_A[
    generates TS_3//Timestamps_2
    knows private K_A_ii //the new extracted secret-key
    //encrypt the new secret-key with the old one
    secret=ENC(K_A_i,K_A_ii)
    msg_3=HASH(TS_3,K_A_i,K_A_ii)//Calculate msg_3 integrity control
]//Send the aythentication data to the server
IoT_device_A -> Trusted_Server: TS_3,msg_3,secret // (3)
principal Trusted_Server[
    //decrypt secret and recuperate the new secret-key
    K_A_ii_ = DEC(K_A_i,secret)
    //check msg_3 integrity (Server authenticated ?)
    _ =ASSERT(HASH(TS_3,K_A_i,K_A_ii_),msg_3)?
]queries[//check mutual authentication
authentication?  IoT_device_A → Trusted_Server: msg_1
authentication? Trusted_Server → IoT_device_A: msg_2
authentication?  IoT_device_A → Trusted_Server: msg_3
confidentiality? ID_a
confidentiality? K_A_i
confidentiality? K_A_ii
freshness? msg_1 freshness? msg_2 freshness? msg_3
authentication?  IoT_device_A → Trusted_Server: secret
]
```