T2S-MAKEP and T2T-MAKEP: A PUF-based Mutual Authentication and Key Exchange Protocol for IoT devices

Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa

T2S-MAKEP and T2T-MAKEP: A PUF-Based Mutual Authentication and Key Exchange Protocol for IoT devices

Abstract*—Nowadays, more constrained devices are becoming connected, building an extensive Internet of Things (IoT) network, but suffering from many security issues. In particular, authentication has become a severe research challenge for IoT systems. Furthermore, confidentiality, integrity, and availability are considered the core underpinnings of information security in general. Unfortunately, deploying conventional authentication protocols for IoT devices in practice is challenging for two main reasons. First, IoT devices have limited memory capacity, processing power, and energy resources. Second, these protocols store secret keys in the IoT devices' volatile memory, making them vulnerable to physical attacks. Luckily, Physical Unclonable Functions (PUF) has emerged as promising low-cost security primitive. A PUF eliminates the need to store secret keys in device memory, making it a potential alternative to deploying more secure and low-cost authentication protocol schemes for IoT systems. Thing-to-Thing (T2T) or direct connection between IoT devices represents a promising technique to enable things to communicate directly without the need for a trusted third party. This paper proposes two novel lightweight Mutual Authentication and Key Exchange Protocols (MAKEP) for IoT devices using PUF. The first scheme, called T2S-MAKEP, ensures secure communication for Thing-to-Server (T2S). The second, called T2T-MAKEP, allows two endpoints of resource-constrained IoT devices, each with an embedded PUF circuit, to communicate securely. Both proposed protocols, T2T-MAKEP and T2S-MAKEP, allow for robust authentication without storing any information on the device's memory and simultaneously establish the session key exchange. Our proposed protocols have been verified and validated using the automatic security analysis checker, Verifpal.

***Index Terms*—IoT Security, PUF-based authentication, Thing-to-Thing authentication, Key Exchange Protocol, Fuzzy Extractor.**

I. Introduction

NOWADAYS, equipment and devices are rigged with network connectivity, sensors, and a wide range of communication protocols and techniques to communicate with each other, constituting a so-called Internet of Things (IoT). Such devices are widely used in many applications, starting from smart homes to public health. Wireless technology is the most means of communication used by these devices to transfer a large amount of data, making them a prime target for cyber-attacks. Further, IoT devices face several security issues that must be continuously managed and maintained, including authentication, privacy, access control, and data collection and management [24]. However, ensuring secure communication between these devices requires a robust authentication protocol that refers to verifying identities and preventing malicious ones from accessing the trusted IoT network and a secure session key for securing transmission after a successful authentication phase. Unfortunately, any defect in the authentication protocol allows an unauthorized thing to communicate, inject false data, get access to confidential data, and launch dangerous attacks with other things.

Generally, traditional symmetric or asymmetric cryptographic algorithms require more processing power, large memory, and high energy sources to secure authentication and communication. Nevertheless, the limitations in memory capacity, processing power, and energy resources of the IoT devices impede deploying conventional authentication protocols in IoT networks. Further, IoT devices can be found in public areas, and the conventional schemes store the secret keys on the local volatile memory of the device, making them vulnerable to physical attacks [29]. In such a situation, an adversary can easily access physically to the IoT device and recuperate the stored secret information or even clone the thing's embedded circuits. To overcome these limitations and issues, recently, Physical Unclonable Functions (PUF) became a hot topic in research and development by exploiting the physical disorder of physical things. PUF can generate unique secret information from the physical characteristics of the IoT device and use it as a unique device fingerprint, making PUF a very efficient solution for IoT authentication protocol.

In practice, Thing-to-Thing (T2T) and Thing-to-Server (T2S) are two different authentication scenarios. In T2S, the thing wants to authenticate and communicate with the trusted server in the network. Besides, T2T or direct IoT end-devices connection represents a promising technique to enable things to communicate directly without the server's interaction. This work proposes a novel lightweight (T2S, T2T) Mutual Authentication and Key Exchange Protocol for IoT devices (T2S-MAKEP, T2T-MAKEP) using PUF.

A. Related work

Over the past decade, a substantial amount of research has focused on the authentication and key exchange of IoT devices. Specifically, extensive investigation has been carried out on PUF-based authentication protocols for IoT applications. These protocols utilize various types of PUFs, employ different authentication mechanisms, and aim to offer a lightweight and secure authentication scheme under diverse settings such as smart grid, IoMT, and IoT.

Most of them are useful for authentication between an IoT device with a trusted server [1, 9, 11, 14, 15, 20, 21, 22, 23, 25, 27], and some protocols take into consideration a thing-to-thing authentication and the key exchange [2, 3, 32]. In this section, we briefly review some recent IoT PUF-based authentication protocols.

Idriss et al.. [11] proposed a lightweight PUF-based protocol that offers mutual authentication for IoT devices [11]. Instead of storing the challenge-response pairs (CRPs) on the server, they save the CRPs trained soft model while challenges are exchanged during the authentication process. This protocol does not ensure communication reliability, especially error correction. Following the same schema of using PUF without storing many CRPs on the server, [14]. [14] proposed a two-way authentication protocol by relying on the PUF of a device, where only one pair of CRPs is stored on the server and for each successful authentication phase is ended by the generation and storage of a new CRP pair. This protocol is vulnerable to physical attacks since it stores the used secret response on the device rather than other variables that are classified as secret. This scheme could be improved by resuming some unused variables and enhancing its reliability. Yanambaka et al. [27] presented a PUF-based authentication scheme for the Internet of Medical Things (IoMT) where both the server and the IoMT are equipped with a PUF. The authors used a secure database as a third party to store collected CRPs. However, the exchanged messages between the IoMT and the server has not been subject to any encryption or camouflage techniques that facilitate easily launching modeling attacks. Also, error correction has not been taken into consideration. Zheng et al. [32]. proposed a lightweight PUF-based mutual authentication and a key-exchange protocol between IoT devices (Peer-to-Peer) without the need of the trusted server, except in the setup phase. The protocol was designed for low-cost and resource-limited devices, but awkwardly it needs to store four variables for each device that wants to connect with it. Some of this information is classified as private, making them vulnerable to physical attack. This protocol is not practical because if a new device wants to communicate with an already in-service device, Both have to go through a setup phase with the server. Najafi et al. [23] presented a PUF-based authentication protocol using a Convolutional Neural Networks (CNN) as an alternative solution of using error-correction techniques. This protocol does not offer mutual authentication, and many attacks are not considered, such as modeling, man-in-the-middle and DoS attacks. Guan et al. [9] proposed an identity PUF-authentication protocol for IoT devices using the PUF outputs and the user password as two-factor authentication. 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.

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. Unfortunately, this protocol is unpractical since the error correction is not considered in a UAV environment known for its dramatic changes. Meng et al. [20] proposed a mutual authentication between IoT devices and the server. They rely on PUF as the main security component in addition to hashing functions and XORing operations. Aman et al. proposed a PUF-based authentication protocol called RapidAuth [1] that resists to physical attacks. With the help of using elliptic curve cryptography (ECC) as second security primitive rather than PUF. This scheme does not consider the noise elimination that plays a main role in the encryption and decryption messages during the authentication steps.

Kim et al. [15] proposed a PUF-based IoT device authentication protocol. They follow the same strategy presented in [25] where the device stores only one CRP pair. In fact, using the response as cryptographic key without correcting errors makes the application of the proposed protocol impossible since the key is generated from the responses. Muhal et al.[22] proposed a PUF-based authentication protocol called PAS (PUF Authentication Scheme). In the enrollment phase of this protocol, both the device and the server store an initial session secret key that will be used in the authentication phase. Further, the device stores the initial session key, which allows physical attacks. Unfortunately, the proposed protocol does not use any error correction and noise elimination technique, making it impractical. 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 and different environments.

Byun [3] propose a PUF-based end-to-end authentication and key exchange protocol (AKE) for IoT devices. in this schema, each end party has a distinct PUF-embedded circuit. Except for the setup phase, which runs in a secure area, the AKE is achieved between two IoT devices without the help of the server. This makes it practical for decentralized IoT networks. The proposed scheme is impractical in a real application since each device needs to be a member of a network has to go first through a setup phase where both devices generate and exchange data used in the authentication phase. Further, it does not fit with the IoT devices constraint like memory and processing due to manipulated and stored data by the device. The protocol is vulnerable to physical attacks and helper data manipulation from the security side. Aman et al. [2] present a light-weight mutual PUF-based authentication protocols for IoT systems. They proposed two schemes: when an IoT device wants to authenticate and communicate with its trusted server and when an IoT device wants to communicate with another IoT device without storing secrets on the local device memory. Although the proposed protocol does not store any information on the IoT, both presented scenarios are impractical since they do not take error correction in the authentication steps. Further, the server helps with the device-to-device authentication steps.

B. Motivations

From the related work results, we have learned and observed that:

- When IoT devices store secret or sensitive information used in the authentication process, they are susceptible to physical attacks where an advisory could steal the stored secret information. Furthermore, if the trusted server stores multiple CRPs, it could increase the risk of modeling attacks when an attacker gains access to the

server. On the other hand, if the challenge and response are transmitted in plain text, the protocol becomes vulnerable to machine learning attacks. Additionally, we have observed that:

- The existing protocols did not consider thing-to-thing T2T authentication operation, and most of those that achieve T2T rely on real-time or semi-real-time participation of the server. Moreover, T2T protocols that don't involve the server's participation are vulnerable to physical attacks since they store secret information on the device. Additionally, many of these protocols are impractical due to their scheme, where each registered thing went to communicate with other things of the network it has to go through a setup phase where the server and both things exchange some information that will be used during the authentication process. This means that in a network composed of n things, and each thing has to store $n-1$ secret information. Furthermore, when a new thing wants to join the network, all the registered things on service need to go through an update phase to add the information of the new device to their local database.
- To ensure security, robustness, and resistance against known protocol attacks, many proposed schemes combine PUF with other cryptographic primitives such as Elliptic Curve Cryptography (ECC). Additionally, some schemes employ techniques like data dissimulation and obfuscation, such as the XORing function, to securely exchange response and other sensitive information used in the authentication process, thus protecting against adversaries.
- Since environmental variations significantly affect the regeneration process of a stable response, they can introduce noise and errors in the PUF's output. Therefore, any PUF-based authentication protocol must include a noise elimination and error correction process. One commonly used solution for this problem is the Fuzzy extractor technique, which utilizes public helper data variables during the noise elimination phase. However, storing these variables in the local device memory makes them vulnerable to manipulation attacks targeting the helper data.
- Many authors fail to capitalize on the unique feature of the PUF's response, which enables the generation of a session key from the CRP. This session key can be utilized for future communication without requiring the involvement of an additional entity to deliberately determine the session communication key. This can be achieved at the end of each successful authentication phase.

C. Outlines

The reminding parts are depicted as follows: Preliminary knowledge is presented in Section II, including Physical unclonable function and fuzzy extractor. Then, our contributions and the defined system model and security model are presented in sections III and IV respectively. Our proposed authentication schemes are elaborated in the next section. In Section VIII, we give the formal and informal system security analysis. In addition, we compare and discuss our proposed protocols with the reviewed works. Finally, the conclusion and future work are given in Section IX.

II. Preliminaries

A. Physical Unclonable Functions

The PUF is a one-way function that exploits the unique random imperfections found in the nano-scale level of the structure of physical objects [10]. It is used to extract a piece of unique information called the *response* from the physical characteristics of the object that can be used as a unique identity of the subject or as a cryptographic key. Due to the randomness found on the conventional integrated circuits caused by the manufacturing process variation, silicon PUF [18] refer to PUF built using silicon chips. The Arbiter PUF is the first and the most used silicon PUF, introduced by Gassend et al.. In order to output the response, a PUF needs an input called a challenge. When a challenge stimulates a PUF circuit, the latter will interpret the challenge in the PUF internal system using the complex physical function that is unique to each PUF. Then, the PUF will output a response that is unpredictable but repeatable. A couple of challenge and the correspondent response forms CRP, representing the unique identifier "fingerprint" of the IC or the device where it is embedded on [6]. PUF has to produce the same response for the same given challenge, and as the PUF is a one-way function, it is impossible to predict a response from the challenge or vice versa. Rather than, each PUF instance always produces different responses for the same challenge compared to other instances [19].

In addition to the manufacturing process variation, which makes the birth of SPUF, the variations in the environmental conditions such as temperature, power supply, and aging have a significant impact on the stability of the SPUF outputs by causing noise on their outputs [10, 12]. Hence, the noise can cause one or more errors for the same challenge, resulting in an incorrect response that differs from the initial generated one [31]. Therefore, the new response could not be used directly as a cryptographic key. On another side, the cryptographic key may be used several times, so it must have the possibility of the regeneration process of the same key. This problem can be solved using the fuzzy extractor (FE) [4] technique, an error correction solution to regenerate stable responses from PUF.

B. Fuzzy Extractor

Fuzzy Extractor (FE) introduced in [4] is designed for extracting nearly uniform random string from noisy and non-uniform random data with high entropy. FE is built from a pair of algorithms to extract stable, reproducible information from the PUF response; generation (Gen) and reproduction (Rep). Gen takes the initial response and outputs uniform random string data (refer to the cryptographic key) and non-secret data called public helper data. In order to reproduce the key from a noisy response, the reproduction algorithm takes two inputs: the noisy response and the public helper data. The reproduction succeeds only if the initial and noisy

responses are close enough. As shown by Figure 1, given the same challenge C_1 as input to the same PUF module PUF_1, in different temperatures $T_1 = 30K$ and $T_2 = 80k$, the PUF generates two different responses R_1 and R'_1. We consider the first response the initial, whereas the second is the noisy response. We use the Gen procedure to generate the secret key k and the public helper data P. Then, for the reproduction of the same key, we use the Rep procedure, which takes the noisy response and P as input [28].

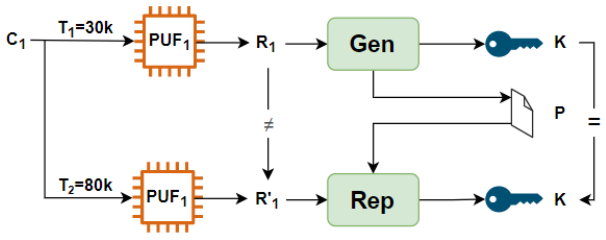

Fig. 1. Typical scheme for a fuzzy extractor.

C. PUF-based Authentication

As shown in Figure 2, PUF-based authentication protocols can be accomplished through two distinct phases. Firstly, during the enrolment phase, the server has secure access to the IoT device, applies a set of challenges, and then stores their corresponding responses extracted from the PUF circuit integrated within the device. The second phase is verification, in which the server verifies the identity of the IoT thing. Then, 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 received response from the server-side matches the stored one corresponding to the used challenge, then the IoT thing is authenticated and can have access to the IoT network. Otherwise, the authentication fails.

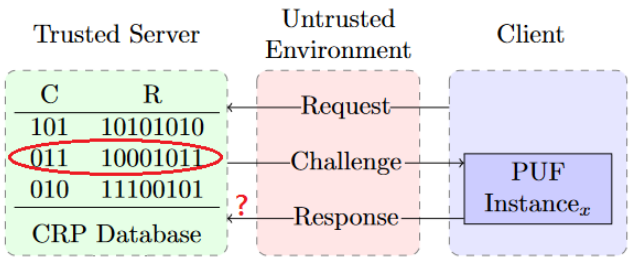

Fig. 2. A PUF-based Authentications Protocol Overview.

III. Our Contributions

In response to the above-discussed limitations of the surveyed PUF-based protocols, in this work, we use PUF as security primitive, where the CRPs of PUF are used by connected things to authenticate themselves. According to the related work, we can find two cases for the IoT devices authentication process: *thing-to-server* and *thing-to-thing*. In the first case, the IoT thing communicates with the server, and in the second, the IoT thing communicates with another thing from the same network. This paper takes both authentication cases by developing the following contributions.

1) We introduce two different PUF-based authentication schemes: T2S-MAKEP for mutual authentication and key exchange protocol between a thing and a server, and T2T-MAKEP for a PUF-based thing-to-thing mutual authentication and key exchange protocol between two things.
2) To resist physical attacks, we develop and validate a protocol scheme that accomplishes authentication without storing any secret or public information on the device's local memory.
3) For robust communication, our authentication mechanism considers error and noise elimination using fuzzy extractor techniques.
4) In addition, our solution allows device-to-device communication without a full intervention of the sever.
5) Our developed protocols are verified and validated through formal using Verifpal. Also, we ensured the resilience of our protocols to the most known attacks to IoT protocols.

IV. System Model & Security Model

This section presents the system and an adversary model for security analysis of the proposed protocols. The system model shows the network architecture where protocols could be appliqued and the network component's requirements to run and use our protocols. The adversary model concerns threats found on the system model, including different attacks that an intruder can run.

A. System Model

As shown in Figure 3, we consider the same system model of IoT as in [2, 17, 30, 32]. The network model mainly contains two entities: Things and the server. Things could include various sensor devices, and the server is responsible for managing things and storing security parameters. Things can communicate with the server, but they can communicate directly with other things through the internet network. Although the considered system model is simple, the proposed protocol can be applied to various complex IoT network models. In this paper, we assume that:

- Each IoT thing is equipped with an embedded PUF circuit.
- Any physical tampering with the PUF will irreversibly modify the slight physical variations in the integrated circuit, which in turn changes the PUF challenge-response behaviour, or even destroys the PUF circuit.
- The IoT things have limited storage capacity and cannot protect any stored secrets in their local storage memory.
- The server is considered the trusted party with no limitation of resources that can be found only in a secure area.
- The communication between the IoT thing microcontroller and its PUF component cannot be accessed only through a secure channel [2].

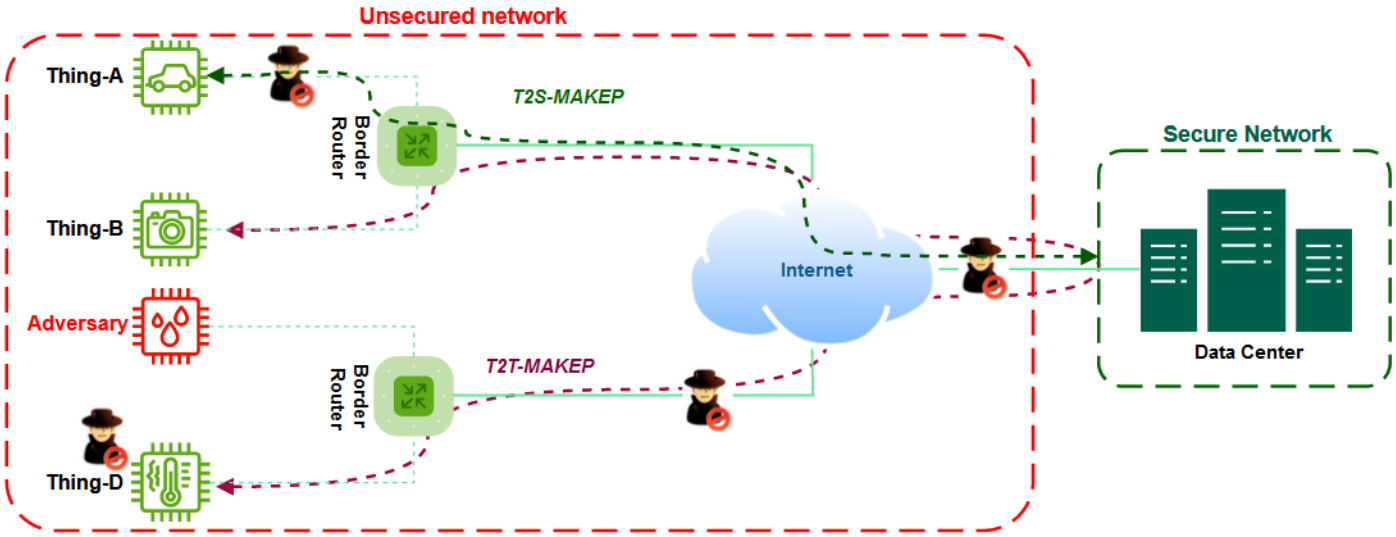

Fig. 3. The system model of the proposed protocols.

- The proposed solution targets single-hop networks.

B. Adversary Model

The proposed protocol is designed keeping in mind the following adversarial model where common assumptions are made. The following assumptions are considered as the capabilities of the adversary.

- The adversary can capture things and obtain the crucial stored values from their memory
- The adversary has complete control over the public channel, allowing them to intercept, revert, modify, replay, or even generate a completely fabricated message
- The Server's attached database is currently not accessible, and as a result, the secret information stored within the database cannot be accessed by any potential adversaries
- By intercepting messages transmitted through insecure, public wireless channels, the adversary is able to acquire knowledge of the messages. This knowledge enables the adversary to craft a valid message, which can then be inserted or modified as needed
- The adversary can disrupt the network using a denial of service attack
- The server situated in a highly secured environment providing hardware/software network security solutions like IDS/IPS, firewalls, anti-flooding and anti-DoS, etc.

V. T2T-MAKEP Scheme

This section presents the proposed Mutual Authentication and Key Exchange Protocol for IoT devices equipped with a PUF. Our protocol uses PUF for both IoT devices authentication cases, T2S and T2T, which need an initial or setup phase. In addition to using PUF as a source of cryptographic key generation, the fuzzy extractor is also considered to reproduce the same keys several times and hashing functions to uniform the generated keys. Before presenting this step, we provide in Table I the main notations used in this protocol.

TABLE I
Authentication protocol's symbols.

Symbols	Definitions
ID_A	The identity of an IoT thing 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 the 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	Timestamp
$\|\|$	Concatenation symbol

A. Setup phase

Contrarily to the most existing PUF-based authentication protocols that store a considerable number of CRPs, our authentication mechanism stores only one pair. Thus, it minimizes the security threat due to the confidentiality of the stored data and resources of the server database [15]. When a new device needs to be added as a member of the network, it goes first with the server through the setup phase that is executed in a trusted and secure environment. As presented in Figure 4, the setup phase should respect the following steps to perform a secure connection between $Thing-A$ and $Trusted-server$.

- $Thing-A$ 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 $Thing-A$ within ID_A.
- $Thing-A$ inputs this challenge, $C_{A,i}$, into its PUF component to output the corresponding response $R_{A,i} = PUF_A(C_{A,i})$. Then, $Thing-A$ sends to the server: ID_A and $R_{A,i}$.
- Using the generation procedure of fuzzy extractor Gen, the $Trusted-server$ extracts the secret key $K_{A,i}$ and the

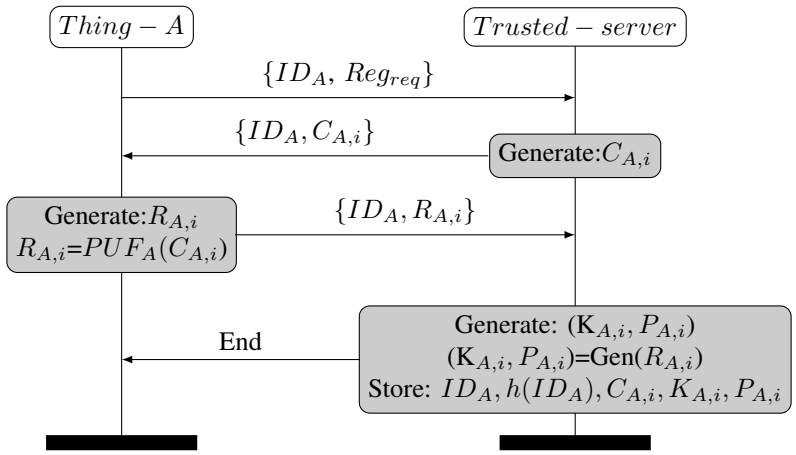

Fig. 4. T2S-MAKEP's setup Phase.

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 stores $ID_A, h(ID_A), C_{A,i}, K_{A,i}, P_{A,i}$ on its local secure database.

- At the end, $Trusted - server$ informs $Thing - A$ about the end of the registration/setup phase.

B. T2S-MAKEP Protocol

To show how a thing can be identified and authenticated after its successful registration in a secure environment, we present first our proposed mutual authentication scheme between the IoT device and the server, called T2S-MAKEP. The thing and the server achieve mutual authentication since they are only the entities that know about the generated secret key for a given challenge in the setup phase and stored on the trusted server. 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 device does not store any secret and public information, which avoids physical attacks.

Our scheme compares the stored response in the setup phase with the new generated one for the same given challenge to check the device's identity. Unfortunately, generating the same response for the same challenge in different environments and conditions such as voltage and temperature makes the response noisy and 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 considers the noise elimination process using the fuzzy extractor.

On the other side, the server and the IoT device exchange securely a session key that corresponds to the used secret key. It is used to protect and secure the communication between things and the server. As shown in Figure 5, the authentication process between the IoT device (Thing-A) and the server (trusted-server) is running in three steps as follows.

In **Step (1)**, Thing-A generates a timestamp TS_1 and calculates a hash value $m_1 = h(ID_A, TS_1)$ of its identity ID_A and TS_1. Then, it sends the hash of its identifier $h(ID_A)$, the authentication request $Auth_{req}$, TS_1 and m_1 message to the server.

In **Step (2)**, upon receiving the message from the IoT device, the server checks in its database if the received $h(ID_A)$ exists. If it does not, the server rejects the authentication request. Else, 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 was not corrupted or tampered during the transmission phase. To calculate the message $m_2 = h(ID_A, C_{A,i}, P_{A,i}, K_{A,i}, TS_2)$, the server retrieves $C_{A,i}, P_{A,i}, K_{A,i}$ that belongs to the ID_A from its database to its memory, and it generates a timestamp TS_2. Finally, the server sends the stored challenge $C_{A,i}$, the helper data correspondent to this challenge $P_{A,i}$, TS_2 and the message m_2 to $Thing - A$.

In **Step (3)**, once $Thing - A$ receives the server response, it regenerates the response proper to the received challenge $R'_{A,i}$ = $PUF_A(C_{A,i})$ that is considered noisy. Then, it reproduces the secret key $K_{A,i}$ from the noisy response $R'_{A,i}$ using the fuzzy extractor reproduction process $K_{A,i}$= $Rep(R'_{A,i}, P_{A,i})$. To verify the integrity of the received message from the server, it calculates $h(ID_A, C_{A,i}, P_{A,i}, K_{A,i}, TS_2)$, and compares it with the received m_2 message. This provides the first factor for authenticating the server, where $Thing - A$ verifies the authenticity of the server based on the success of matching the messages m_2. This can be successful because 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 corresponding response of the new computed challenge $R_{A,i+1}$=$PUF_A(C_{A,i+1})$. Finally, the device calculates $m_3 = h(ID_A, TS_3, k_{A,i}, R_{A,i+1})$, encrypts the new response with the reproduced secret key $m_4 = (R_{A,i+1})_{K_{A,i}}$, and sends back the calculated messages m_3 and m_4 with TS_3.

Upon receiving the message from $Thing-A$, the server decrypts m_4 using the stored secret key $K_{A,i}$, and verifies the integrity of the received data by calculating m_3. Then, it compares the received and the calculated hash messages. If the matching fails, the server terminates the connection. Else, the server verifies the authenticity of the IoT device as a successful matching of these two hash messages. It means that the IoT is authenticated, and the server communicates with the right IoT device. In this step, the server and the IoT device can use the secret, and the exchanged key $K_{A,i}$ as session key to secure communication during the current session.

After a successful authentication of the IoT device, the server generates a new secret key and its corresponding helper data from the new generated response $R_{A,i+1}$, using the fuzzy extractor generated procedure, $(K_{A,i+1}, P_{A,i+1}) = Gen(R_{A,i+1})$ and calculates the new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$. Finally, the server replaces the used information $C_{A,i}, P_{A,i}, K_{A,i}$ by the new one $C_{A,i+1}, P_{A,i+1}, K_{A,i+1}$ to be used in a future authentication process.

C. T2T-MAKEP Protocol

In this section, we detail the proposed mutual authentication and key exchange protocol between two things (Thing-A and Thing-B). By following T2T-MAKEP rules, things could authenticate and communicate each to others without a real participation of the trusted server. As shown in Figure 6, T2S-MAKEP is used as an essential step at the first stage of T2T-MAKEP executions that are detailed as follows.

When an IoT end-device $Thing-A$ wants to communicate with another end-device $Thing-B$, first in **Step (1)**, $Thing-A$ generates a timestamp TS_1 and calculates a message m_1, where $m_1 = h(ID_A, h(ID_B), TS_1)$ to ensure the integrity of the T2T communication request message. Then, the IoT device sends the hash of its identifier $h(ID_A)$, the hash of $Thing-B$ identifier $h(ID_B)$, the authentication request Com_{req}, the message m_1 and TS_1 to the server.

In **Step (2)**, upon receiving the message from $Thing_A$, the server checks the validity of $Thing_B$ identity by checking $h(ID_B)$ in its database. If the finding fails, the server rejects the authentication request. Otherwise, the server follows steps of Section V-B to launch T2S-MAKEP process and verify the authenticity of the IoT device lunched the communication request ($Thing_A$).

At the end of the T2S-MAKEP phase, if the authentication fails, the server rejects the authentication request. Else, $Thing-A$ is authenticated and the message of the communication request was not corrupted or tampered during the transmission phase. Then, the server retrieves the correspondent stored data of $Thing-B$: ID_B, $C_{B,i}, P_{B,i}$, $K_{B,i}$ and generates a timestamp TS_2.

After that, the server calculates both messages m_2 and m_3, where $m_2 = h(ID_A, C_{B,i}, P_{B,i}, K_{B,i}, TS_2)$ which is used to guaranty the integrity of the transmitted data. In addition, $m_3 = (K_{B,i})_{K_{A,i}}$ is a result of the cryptographic operation on the secret key $K_{B,i}$ using the secure session key $K_{A,i}$ established at the end of T2S-MAKEP phase. Finally, the server sends $C_{B,i}$, $P_{B,i}$,TS_2,m_2 and m_3 to the device that lunched the communication request $Thing-A$.

In **Step (3)**, once $Thing-A$ receives the server response, it decrypts m_3 by using the T2S-MAKEP session key $K_{A,i}$. Then, it verifies the integrity of the received response by calculating m_2 using the decrypted message $K_{B,i}$ and the non-encrypted received data ($C_{B,i}$, $P_{B,i}$, and TS_2). If the verification fails, the connection is rejected. Else, $Thing-A$ generates a timestamp TS_3 and calculates $m_4 = h(C_{B,i}, P_{B,i}, K_{A,i}, TS_3)$, $m_5 = (K_{A,i})_{K_{B,i}}$ and $S = h(K_{B,i} \oplus K_{A,i})$ which is the new session key used to secure the communication between the two things by Xoring the secret keys of both devices. Finally, $Thing-A$ sends the hashed identity of $h(ID_B)$, the communication request Com_{req}, the challenge $C_{B,i}$, the helper data $P_{B,i}$, a timestamp TS_3, the control of the integrity m_4 and the encrypted message m_5.

Afterwards,$Thing-B$ uses its PUF component and generates the response of the received challenge $R'_{B,i} = PUF_B(C_{B,i})$, which is considered as a noisy response. Then, it reproduces the original secret key $K_{B,i}$ from the noisy response through the fuzzy extractor reproduction process $K_{B,i} = Rep(R'_{B,i}, P_{B,i})$. It uses this key to decrypt the received encrypted message m_5, then the result $K_{A,i}$ is used to calculate and verify the integrity of m_4. If the verification passed, the IoT end-device $Thing-B$ makes sure that the message was not tampered during the transmission phase. Also since only the server has a secret key $K_{B,i}$, the device requesting the connection is trusted with the same trusted network and server. Finally, $Thing-B$ calculates the new session key $S = h(K_{B,i} \oplus K_{A,i})$ used to secure the communication with $Thing-A$. At this step, both IoT end-devices could communicate securely using the secret session key S.

Finally, in **Step (4)** $Thing-B$ and the server execute the needed operations to update the used information $C_{B,i}, P_{B,i}, K_{B,i}$ by the new generated one $C_{B,i+1}, P_{B,i+1}, K_{B,i+1}$

VI. Security Analysis

In this section, we first introduce briefly various attacks against IoT PUF-based authentication protocols. Then, we present the informal and formal security analysis of our proposed protocols (T2S/T2T-MAKEP). For the informal security analysis, we check the robustness of the proposed protocol against the presented attacks. Formally, we analyze our proposed protocols using the automatic security verification checker, Verifpal [16].

A. Attack Scenarios

A brief definition about the most known IoT systems attacks [14, 21] that an attacker can launch on an IoT authentication protocol is presented in Table II.

B. Informal security analysis

Our proposed mutual authentication mechanism is robust against the attacks presented in Section VI-A. This strength comes from using PUF without storing any information (secret or not) in the local memory of the IoT end devices. Rather

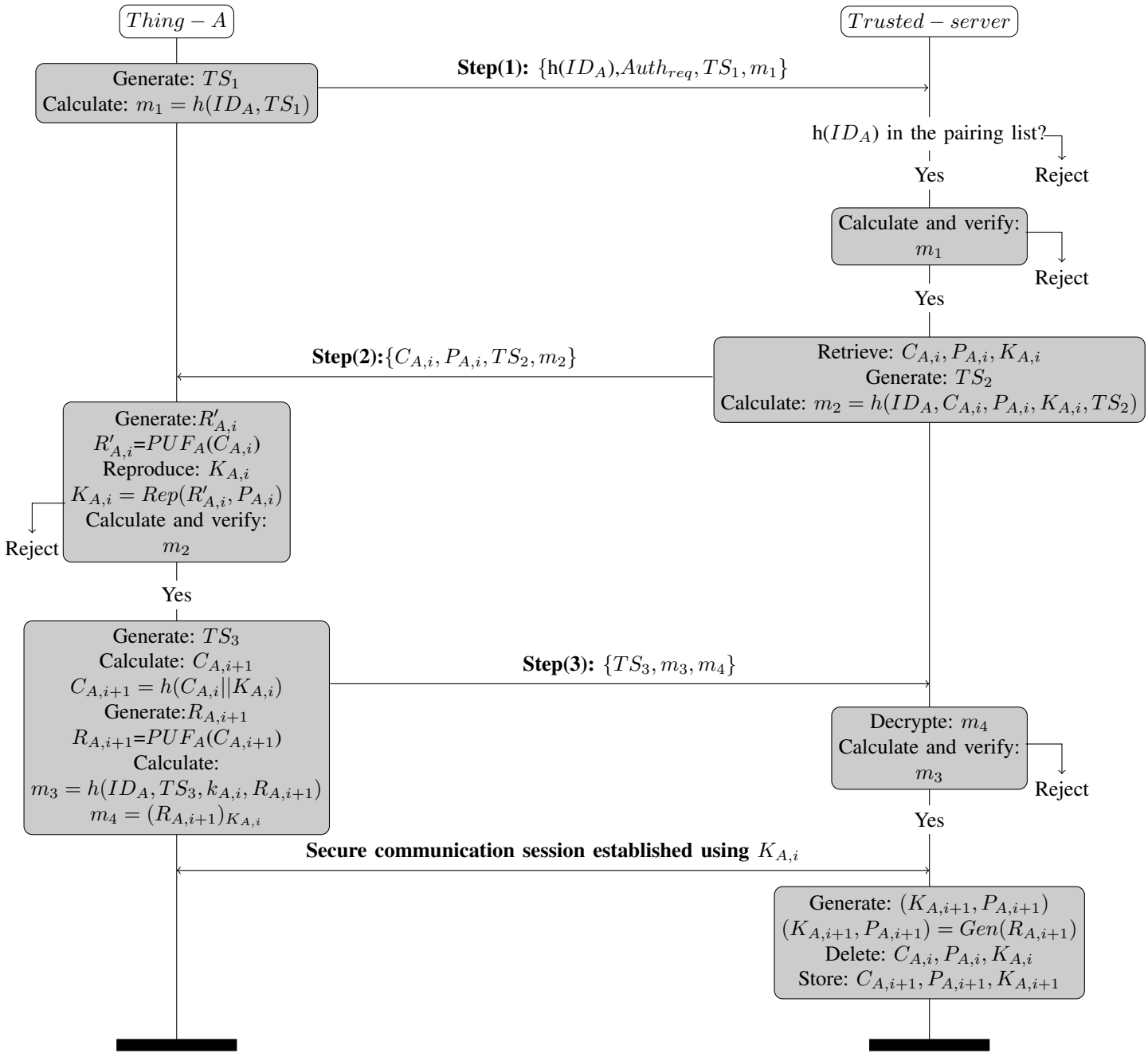

Fig. 5. T2S-MAKEP protocol's scheme.

than, no sensitive information is transmitted in clear during the authentication process of both proposed protocols. A defence assessment of each presented attack scenario is given in the following points.

- *Resisting to message analysis attack.* In both schemes, secret keys, session keys, and responses are confidential and cannot be accessible by an attacker despite the fact the possibility of intercepting the authentication messages. This is achieved by encryption and hashing the transmitted messages and not storing them locally.
- *Resisting to replay attack.* By considering an attacker was able to intercept old authentication messages and want to replay them, she/he cannot forge the current transmitted message since a valid timestamp is assigned at each transmitted message. Further, except for the identity of IoT end devices, all parameters are updated after each new session. Consequently, replay attacks are prevented efficiently.
- *Resisting to denial of service attacks.* Only two types of entities could be found in both proposed protocols, a server and the IoT end-device. However, the DOS attack is infeasible on the server side due to its high computation power [14]. So, this attack is considered only on the ToT end-device side. The attack can be done in three cases:

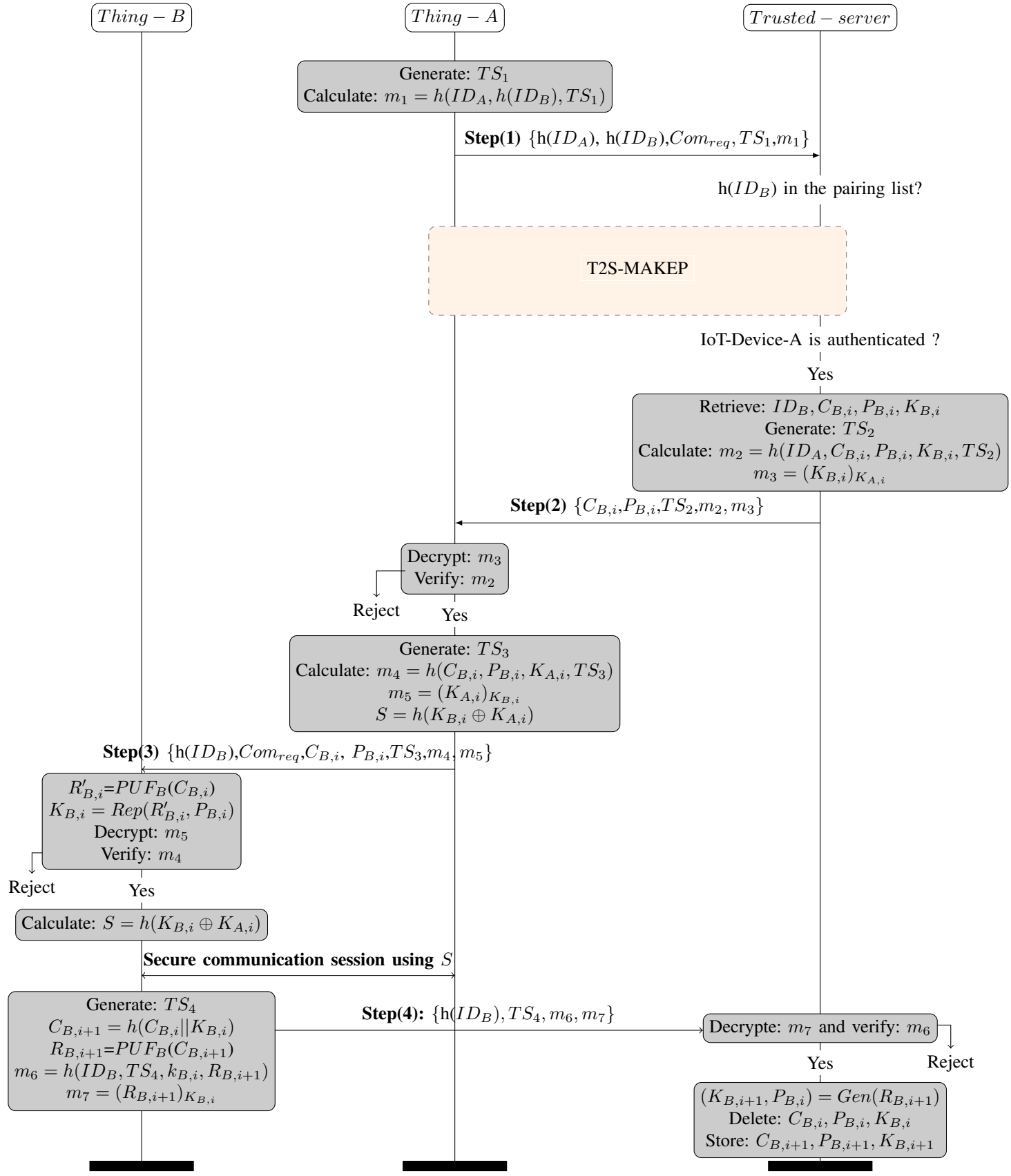

Fig. 6. T2S-MAKEP protocol's scheme.

TABLE II
ATTACK SCENARIOS.

Attack Scenario	Definition
Message Analysis	An attacker tries to intercept the transmitted information between the communication entities.
Replay	The attacker stores the transmitted information of the valid authentication operation to exploit them in a future authentication.
Denial of Service	An attacker tries to disrupt temporarily/indefinitely the IoT device's services by flooding the targeted device with an overflow of authentication requests.
Physical attack	An attacker attempts accessing physically to the device and obtain the stored secret.
Man-in-the-middle/Injection	An attacker injects data and changes the transmitted messages between authenticated devices.
Helper data manipulation	Any manipulation of the helper data by an attacker caused errors on the result of the response's reproduction.
Modeling	Collecting CRPs datasets and constructing machine learning techniques to offer the possibility of predicting the correct response related to a given challenge.

in **step (2)** for the T2S-MAKEP scheme and **step (2)** and **step (3)** for the T2T-MAKEP scheme. In each step, after receiving the message by the device, it checks the integrity of the received packet by computing its one hash function. Further, the probability that the random guessing of the hashing value to pass the verification process is negligible due to the confidentiality of the secret key included in each hashed value. As a result, the Dos attack is not practical in our scheme.

- *Resisting physical attack.* In both schemes, IoT end-devices, do not store any secret or sensitive information in their local memory. Further, one of the assumptions of the target system model is that the communication between the PUF circuit and the IoT device's micro-controller is considered secure. Thus, making our proposed protocols secure against physical attacks, even if An attacker captures ioT end-devices.
- *MITM and Injection attack:* All the authentication messages do not include any secret or sensitive information in plaintext in our proposed protocols. The only exchanged data in plaintext during authentication are the thing's identity, timestamps, challenges and public helper data. Also, any secret is hashed or encrypted using a one-way hashing function or a generated secret key known only by the authentication entities. A MITM attack will not benefit from any captured data in such settings. Thus, the proposed protocol is secure against this type of attack.
- *Modeling attack.* This type of attack is based on capturing a set of CRPs. In our mechanism, only the challenge was communicated from the server to the IoT end-devices in plaintext. So, even if an attacker could intercept the authentication's messages, she/he will see only the challenge, which will be useless for launching a modelling attack. This mechanism makes this attack not applicable in our proposed schemes.
- *Helper data manipulation:* In our mechanism, the helper data is securely stored on the server and transmitted to the IoT end-devices in plaintext to reproduce secret keys. So, this parameter can be manipulated only during its transmission. Further, each helper data is transmitted with a hashed message of helper data and other additional information. The hashed value guarantees that the public helper data was not manipulated during the transmission. So, this attack is not feasible in our protocols.

C. Formal security analysis

In order to analyze our proposed protocols formally, we have selected Verifpal, a modern tool dedicated to the formal verification of the security of cryptographic protocols. It is inspired by some older formal verification tools such as ProVerif, and it covers the most well-known model, Dolev-Yao, which is a modelling technique for verifying cryptographic protocols [5]. To model and check how much our developed protocols are secure using Verifpal, we must first define whether our schemes will be analyzed under a passive or active attacker. A passive attacker can observe the communication messages over the network and cannot inject or modify these messages. Contrarily, an active attacker can observe, modify, and inject new messages. In our case, we chose the active one who can alter the authentication messages and inject its data. Secondly, we have to define the different principles (devices and the trusted server) taking part in our communication system, including the active attacker. Then, we describe the authentication messages being communicated between the model's parts across the network. Finally, we query Verifpal about what we would like to test, for example, the confidentiality of the exchanged data and the authentication of the participants.

1) Verifying Security of T2S-MAKEP Protocol: Here, we use Verifpal to verify the security robustness of T2S-MAKEP by checking its possible vulnerabilities during the authentication phase, especially the authenticity, confidentiality, and freshness of the exchanged messages. The first proposed protocol introduced in Section V-B is translated into Verifpal scripts and evaluated using Verifpal's queries.

- **Reachability and Secrecy:** The first objective of T2S-MAKEP protocol is to guarantee the reachability and secrecy of the shared secrets between the IoT end-device and the trusted server. Using Verifpal, we show that the considered secrets id_a, h_{ka1} and r_{a2} are secretly shared between $Thing - A$ and the server (i.e. these secrets cannot be obtained, deleted or altered by an active attacker). To test this property, the following variables are declared.

$$\textit{knows private } id_a$$
$$\textit{knows private } h_{ka1}$$

knows private r_{a2}

knows private is a Verifpal's predefined primitive used to declare private information. The first variable is declared private to guarantee the anonymity of Thing-A identity. The second is the secret and the session key. The last one is the new generated response used to generate the new secret key that will be used for future authentication. At the end of the model, we ask Verifpal for the next confidentiality query of the privates used information. The confidentiality query is the most basic of all Verifpal queries, which are used to test if an attacker could obtain the private information communicated between $Thing-A$ and the server during the authentication phase. Those queries are expressed as follows to check the confidentiality of id_a, $h\_k\_a\_1$, and r_a_2.

confidentiality? id_a
confidentiality? $h\_k\_a\_1$
confidentiality? r_a_2

- **Mutual Authentication:** The second objective of the proposed T2S-MAKEP scheme is the mutual authentication between the IoT device and the server, which adds more resistance to the protocol against man-in-the-middle and replay attacks. To test this security property under Verifpal, we use the following two authentication queries:

authentication? Trusted_server $\longrightarrow$ *Ting_a: msg1*
authentication? Thing_a $\longrightarrow$ *Trusted_server: msg2*

The first query checks if $Thing_a$ can authenticate the $Trusted\_server$ based on the received message $msg1$ in step (2) of T2S-MAKEP scheme. This operation succeeds because $msg1$ results from a one-way hash function, which contains the secret key h_{ka1} known only by the server and is communicated in the setup phase. So, this secret information guarantees that $msg1$ comes from an authentic entity, the $Trusted\_server$. Also, in the second authentication query, the server authenticates $Thing_a$ based on the value of the secret key that is reproduced using FE on the noisy generated response. This step succeeds because only $Thing\_a$ could use its PUF to regenerate the same stored key corresponding to the stored one on the server. Both queries guarantee mutual Authentication.

- **Freshness:** In addition to the above-specified security properties, freshness of the transmitted messages is also a pillar requirement. Freshness query is useful for detecting replay attacks, where an attacker could manipulate one message to make it seems valid in two different contexts. To assert this property, the following Verifpal queries are declared, where each tested message contains a timestamp value that guarantees the freshness of data.

freshness? msg1
freshness? msg2
freshness? msg3

As shown in Figure 7, our proposed T2S-MAKEP scheme ensure mutual authentication between the server and the IoT end-devices. It also proves the reachability and secrecy of the shared keys and information. In addition, it satisfies the freshness of messages during communication.

```
Verifpal • Verification completed for 'T2S-MAKEP.vp' at 10:14:21 AM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
```

Fig. 7. T2S-MAKEP's Verifpal code execution result.

2) Verifying Security of T2T-MAKEP Protocol: This section presents the security analysis for our proposed T2T-MAKEP protocol between two IoT end-devices. This analysis follows the same verification and security properties specification as the same analysis paradigm of T2S-MAKEP. We do not repeat the T2S-MAKEP authentication step between $Thing-A$ and the trusted server to avoid repetitions. Thus, we consider that $Thing-A$ is successfully authenticated, and our analysis started from Step (2) of T2T-MAKEP scheme (6).

```
queries[
  authentication? Trusted_server -> Ting_a: m2
  authentication? Ting_a -> Ting_b: m4
  authentication? Trusted_server -> Ting_a: m3[]
  authentication? Ting_a -> Ting_b: m5[]
  confidentiality? id_a
  confidentiality? s_a
  confidentiality? s_b
  confidentiality? h_k_b_1
  confidentiality? h_k_a_1
  freshness? m2
  freshness? m4
  equivalence? s_a,s_b
]
```

Fig. 8. T2T-MAKEP's Verifpal queries code.

As shown in Figure 9, in addition to the IoT end-device identity id_a, h_{ka1} and h_{kb1} that are secret keys of $Thing_a$ and $Thing_b$ respectively; s_a and s_b are equals calculated session keys between both devices ($s_a = s_b$). After the declaration of these device, we check their confidentiality using the confidentiality query. Further, by using the authentication query, we verify the mutual authentication property between the participants in our scheme.

Based on the exchanged m_2 and m_4, and the encrypted ones m_3 and m_5, we check the freshness of m_2 and m_4 using the freshness query. Then, we verify if both IoT end-devices could calculate the same secret session key S, using the equivalence query between the generated keys (s_a) and (s_b) by $Thing_a$ by $Thing_b$, respectively.

Figure 9 shows that T2T-MAKEP has successfully achieved mutual authentication between the IoT end-devices. Further, the confidentiality of secret information and the freshness of the messages are guaranteed. In addition, the generation of the same and unique secret session key generation is satisfied by both devices.

```
Verifpal • Verification completed for 'T2T-MAKEP.vp' at 02:17:08 PM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
```

Fig. 9. T2T-MAKEP's Verifpal code execution result.

VII. Performance analysis

In this section, we present the performance analysis of our proposed protocols and compare it with the most existing and relevant protocols in the literature, those consider our protocol's features like mutual authentication and error correction. In practice, we propose to implement our proposed mechanism using the SHA-2 (hash standard algorithm) since it is already implemented in well-known security applications such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL). Further, it has been incorporated within a large number of commercial security integrated circuits. In comparison to SHA-3, SHA-2 has a broader range of hardware and software support, making it simpler and faster to implement. Also, we assume the use of 128-bit block ciphers such as CLEFIA [13] and AES [21] for encryption and decryption. First, we evaluate our protocol in terms of its computational complexity.

A. Computational Complexity

The computational complexity of the authentication process is determined by the number of primary operations performed and their frequency. Initially, the fundamental operations of the proposed and compared schemes are hashing (N_H), encryption and decryption (N_E), PUF (N_P), and the error correction (N_F). Figure 5 and Figure 6 can be used to figure out how evaluating the identified criteria.

TABLE III
T2S-MAKEP's computational complexity comparison.

Works	IoT Device	Server
[8]	$7N_H$+$2N_P$+$1N_F$+$4N_E$	$7N_H$+$1N_F$+$3N_E$
[26]	$7N_H$+$2N_P$+$1N_F$+$4N_E$	$7N_H$+$1N_F$+$4N_E$
[20]	$7N_H$+$2N_P$+$1N_F$+$4N_E$	$7N_H$+$1N_F$+$4N_E$
T2S-MAKEP	$4N_H$+$2N_P$+$1N_F$+$1N_E$	$4N_H$+$1N_F$+$1N_E$

Table III lists the primary operations run in the proposed mutual authentication protocol, T2S-MAKEP. And as illustrated in Fig. 10 and Fig. 11, the same operators are required in [8], [26] and [20] schemes to achieve the authentication phase. Table III indicates that they all have the same number of PUF and fuzzy extractor operations. In comparison, the T2S-MAKEP scheme utilizes fewer hash operations. Additionally, by treating the xor operation as a cryptographic operation, our proposed protocol contains only one encryption (device) and one decryption (server) operation, whereas [8]'s scheme contains four in the device side and three in server side. [26] and [20] contain four encryption/decryption in both sides. Thus, the computational complexity of T2S-MAKEP mechanism is lower since hashing, encryption, and decryption operations in the others schemes are higher than in our proposed one.

On the other hand, Table IV shows the main operations carried out by IoT devices to achieve end-to-end authentication. We compare the proposed T2T-MAKEP's protocol with [32] and [2] , which allow device-to-device authentication. Regarding error correction consideration, we can still conclude that our protocol has a slight advantage compared with the protocol in [2]. In general, the computation costs of T2T-MAKEP's scheme are similar to those of [32]. However, we do not consider the update phase in [32] that grows their

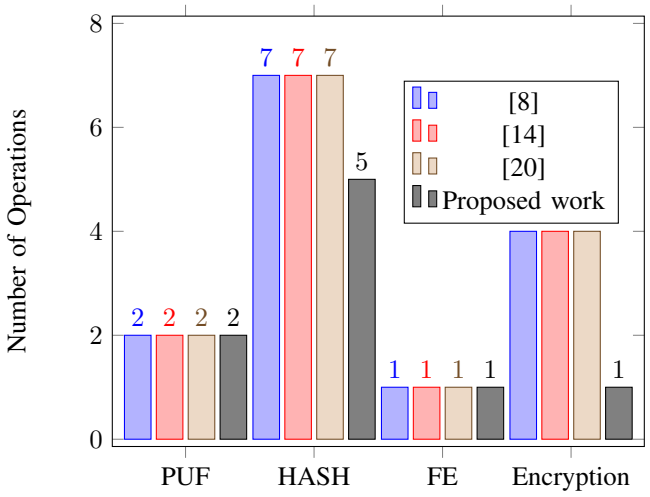

Fig. 10. The Computational Complexity on the device side.

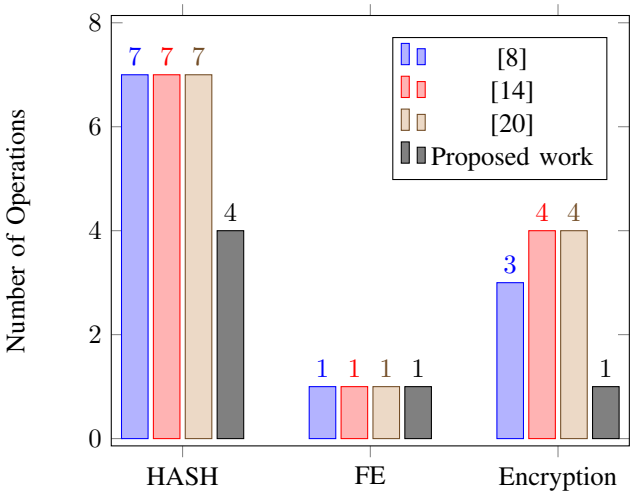

Fig. 11. The Computational Complexity on the server side.

operations considerably, making the computational complexity of the proposed protocols lower than the compared ones.

B. Communication Overhead

Total bytes sent and received during the authentication process is what we mean by "communication overhead". Table VI lists the message parameters and their sizes, which we use to figure out the length of each message that's being sent.

TABLE VI
Authentication messages' parameter values.

Message Parameters	Size in Bits
$Auth_{req}/Com_{req}$	1
Hash function	256
Timestamps	48
Challenge	128
Helper data	64
Encrypted data	128
Device identity	48

Based on the values in Table VI and the protocol steps in Figures 5 and 6, we infer that the transmitted bytes in the T2S-

TABLE IV
T2T-MAKEP'S COMPUTATIONAL COMPLEXITY COMPARISON.

Works	[32]	[2]	T2T-MAKEP
IoT Device A	$4N_H$+$1N_P$+$4N_E$+$1N_F$	$7N_H$+$2N_P$+$4N_E$	$8N_H$+$1N_P$+$4N_E$+$1N_F$
IoT Device B	$4N_H$+$1N_P$+$4N_E$+$1N_F$	$5N_H$+$2N_P$+$3N_E$	$4N_H$+$2N_P$+$3N_E$+$1N_F$

TABLE V
COMPARISON'S SUMMARY BETWEEN T2T-MAKEP AND THE REVIEWED PUF-BASED AUTHENTICATION PROTOCOLS

Work	T2S scheme	T2T scheme	PUF-based authentication	Formal security analysis	Informal security analysis	Mutual authentication	Session key generation	Practicality	Error correction	Thing storage	confidentiality	Freshness
Idriss et al. [11]	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗
Kaveh et al. [14]	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
Yanambaka et al. [27]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗
Zheng et al. [32]	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
Najafi et al. [23]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗
Guan et al. [9]	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
Pu and Li [25]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗
Meng et al. [20]	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Aman et al. [1]	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✓
Kim et al. [15]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗
Muhal et al. [22]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗
Mostafa et al. [21]	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Byun [3]	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
Aman et al. [2]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
T2S-MAKEP	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
T2T-MAKEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

MAKEP scheme are 186 bytes, which is higher than [8] and [26], which have similar transmitted bytes of 150 bytes.On the other hand, to calculate the communication overhead of the T2T-MAKEP protocol, we consider only the transmission between IoT end devices and compare it with the proposed mechanism in [32]. The communication overhead of the T2T-MAKEP protocol is 110 bytes, and 126 bytes for Zheng et al.'s protocol. As a result, the communication overhead in our proposed mutual authentication scheme is lower than Zheng et al.'s mechanism by 12.7%.

C. 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 protocols are very efficient in terms of storage requirements. Since the server keeps only one CRP pair for each device's new authentication and the IoT device does not store any secret or non-secret information in the device's memory. Making our mechanism more scalable for a large number of IoT devices that could be deployed in the system. Contrarily, the IoT devices in [8, 26, 32] have to keep secret information in their local memory.

VIII. COMPARISON AND DISCUSSION

In this section, we run a comparison between our proposed schemes and the existing work reviewed in Section I-A. Our comparison summarised in Table V is based on the following characteristics. ✓ means that the protocol considers the characteristics. However, ✗ means the inverse of the previous statements.

1) *T2T and T2S schemes* to indicate if the considered architecture supports one of both authentication schemes.
2) *PUF-based authentication* shows if the referred protocol uses PUF only as a security primitive to achieve authentication successfully or it needs other computational primitives like the elliptic curve. We do not consider one-way functions, hashing and XORing as complex security primitives.
3) *Formal/informal security analysis* this criteria specify the type of the analysis. Formal relies on sound mathematics approaches to show the correctness of the proposed protocol, whereas informal refers only to a textual description.
4) *Mutual authentication* shows if the protocol supports mutual authentication between all authenticated entities.
5) *Session key generation* indicates if the referred protocol generates a session key to secure the communication after each successful authentication.
6) *Practicality* checks if the protocol is portable, scalable, and robust.
7) *Error correction* indicates if errors and noise in responses are considered.
8) *Thing storage* shows if the IoT device does not store any

secret or non-secret information that helps to accomplish the authentication process.

9) *Confidentiality* indicates if the protocol's secret information is kept confidential during the authentication phase.
10) *Freshness* checks if the exchanged messages are received with respect to precise variables like timestamp, which help not to use an old message in a new authentication.

We found that most of the studied IoT protocols are designed for the T2S authentication protocol scheme, except two [3, 32] that support T2T and only one ([2]) covers both T2S and T2T. From a technical point of view, most of the protocols use one-way and hashing functions to ensure the confidentiality of secret information. Instead of using these two functions, two contributions [1, 9] use another cryptographic algorithm such as Elliptic Curve. In terms of correctness validation and performance evaluation, most of the reviewed protocols do not apply formal and informal security analysis techniques. Regarding the authentication, some of them do not offer mutual authentication [9, 11, 15, 23, 27], nor generate a session key for securing the communication after each successful authentication operation [1, 9, 11, 15, 23, 27]. Further, most of the studied schemes do not consider error correction, which plays a vital role in the practicality of a protocol. Also, for the leakage resilience, we observe that some of the protocols store sensitive information on the local memory of the IoT device. In addition, most of the reviewed protocols ensure the confidentiality of the transmitted secret information but less guarantee the freshness of the transmitted messages.

IX. Conclusion

In this paper, we have developed a thing-to-thing mutual authentication and key exchange protocol for IoT devices (T2T-MAKEP), where a thing wants to communicate with another. First, it has to authenticate with trusted_server using the T2S-MAKEP scheme. In practicality, our schemes guarantee error correction and noise elimination using a fuzzy extractor solution. Further, T2T-MAKEP does not need an update phase without storing any secret or non-secret information on the local memory of the IoT end-device. Also, at the end of each successful authentication phase, our schemes generate a session key and exchange it securely between the communicated entities. In addition, by relying on formal and informal security analysis, we proved that our developed protocols are secure against the most existing attacks, especially physical attacks. Further, it satisfies all pillar security requirements.

In a nutshell, we point out our targets to extend and improve this contribution by the following future works.

- Deploying the proposed protocol with a blockchain architecture that exploits a PUF based on the different nodes devices.
- Considering the energy consumption in our protocol by optimizing the exchanged messages while preserving the communication's reliability.
- Apply the protocol on autonomous vehicles.

References

[1] Aman, M.N., Chaudhry, S.A., 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.

[2] Aman, M.N., Chua, K.C., Sikdar, B., 2017. Mutual authentication in iot systems using physical unclonable functions. IEEE Internet of Things Journal 4, 1327–1340.

[3] Byun, J.W., 2019. End-to-end authenticated key exchange based on different physical unclonable functions. IEEE Access 7, 102951–102965.

[4] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A., 2008. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM journal on computing 38, 97–139.

[5] Dolev, D., Yao, A., 1983. On the security of public key protocols. IEEE Transactions on information theory 29, 198–208.

[6] Gao, Y., Ranasinghe, D.C., Al-Sarawi, S.F., Kavehei, O., Abbott, D., 2016. Emerging physical unclonable functions with nanotechnology. IEEE access 4, 61–80.

[7] Gassend, B., Clarke, D., Van Dijk, M., Devadas, S., 2002. Silicon physical random functions, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 148–160.

[8] Gope, P., Sikdar, B., 2018. Lightweight and privacy-preserving two-factor authentication scheme for iot devices. IEEE Internet of Things Journal 6, 580–589.

[9] Guan, Z., Liu, H., Qin, Y., 2019. Physical unclonable functions for iot device authentication. Journal of Communications and Information Networks 4, 44–54.

[10] Halak, B., 2018. Physically Unclonable Functions: From Basic Design Principles to Advanced Hardware Security Applications. Springer.

[11] Idriss, T.A., Idriss, H.A., Bayoumi, M.A., 2021. A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices. IEEE Access .

[12] Kardaş, S., Çelik, S., Yıldız, M., Levi, A., 2012. Puf-enhanced offline rfid security and privacy. Journal of Network and Computer Applications 35, 2059–2067.

[13] Katagi, M., Moriai, S., 2011. The 128-bit blockcipher clefia. IETF RFC 6114 .

[14] Kaveh, M., Aghapour, S., Martin, D., Mosavi, M.R., 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.

[15] 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.

[16] 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.

[17] Li, S., Zhang, T., Yu, B., He, K., 2020. A provably secure and practical puf-based end-to-end mutual authentication and key exchange protocol for iot. IEEE Sensors Journal 21, 5487–5501.

[18] Lim, D., Lee, J.W., Gassend, B., Suh, G.E., Van Dijk, M., Devadas, S., 2005. Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13, 1200–1205.

[19] Maes, R., 2013. Physically unclonable functions: Constructions, properties and applications. Springer Science & Business Media.

[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, S.J., Peker, Y.K., 2020. Physical unclonable function and hashing are all you need to mutually authenticate iot devices. Sensors 20, 4361.

[22] Muhal, M.A., 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, 2009.

[24] Nandy, T., Idris, M.Y.I.B., Noor, R.M., Kiah, L.M., Lun, L.S., Juma'at, N.B.A., Ahmedy, I., Ghani, N.A., 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] Wang, H., Meng, J., Du, X., Cao, T., Xie, Y., 2022. Lightweight and anonymous mutual authentication protocol for edge iot nodes with physical unclonable function. Security and Communication Networks 2022.

[27] Yanambaka, V.P., Mohanty, S.P., 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, 388–397.

[28] Zerrouki, F., Ouchani, S., Bouarfa, H., 2021a. 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., 2021b. 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., 2022a. Puf-based mutual authentication and session key establishment protocol for iot devices. Journal of Ambient Intelligence and Humanized Computing , 1–19.

[31] Zerrouki, F., Ouchani, S., Bouarfa, H., 2022b. A survey on silicon pufs. Journal of Systems Architecture , 102514.

[32] Zheng, Y., Liu, W., Gu, C., Chang, C.H., 2022. Puf-based mutual authentication and key exchange protocol for peer-to-peer iot applications. IEEE Transactions on Dependable and Secure Computing .